Integrating Inclusive Design and Computing Education

Alannah Oleson

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Program Authorized to Offer Degree:
Information School

University of Washington

Abstract

Integrating Inclusive Design and Computing Education

Alannah Oleson

Chair of the Supervisory Committee:
Amy J. Ko
Information School

To realize more equitable technology futures, it is not enough to simply adapt technology to be more inclusive *after* it is created. We will also need to equip technology creators with the skills they need to critically reflect upon bias and exclusion *during* the technology design process. The question of how to best to impart actionable inclusive design skills to today's computing students—tomorrow's technology creators—remains open. Computing interfaces are an illustrative site of inquiry for demonstrating the concrete impacts of design bias. Interfaces constrain interactions with technology, and by extension, who gets to benefit from technological access and who is excluded. Because technology carries perceptions of objectivity, it can be difficult for students to grasp how subjective design decisions might impact usability. I argue that enabling students to critically reflect upon the ways their design decisions impact users is a key aspect of developing inclusive computing interface design competence.

My work makes four contributions through a series of qualitative and mixed-method studies. First, I contribute a dual-type model of design activity present in computing education contexts, highlighting the need for further investigation into design decision-making skills that help computing students understand the societal impacts of technology. Next, I contribute a set of student learning challenges that arise in introductory interface design courses which may prevent computing students from developing the skills they need to design inclusive technology. Then, I contribute a pedagogical technique that uses a novel strategy called assumption elicitation to help computing students learn to recognize and respond to computing interface design bias, as well as a case study evaluation of its efficacy and considerations for its use in post-secondary design learning contexts. Finally, through an Action Research study of this technique's integration into post-secondary computing courses, I contribute pedagogical content knowledge for teaching inclusive interface design in computing courses, including descriptions of how instructors leveraged the technique to inspire critical reflection, hopefulness, and inclusive design agency in students. These contributions provide foundations for future work in the nascent subdiscipline of critical human-computer interaction (HCI) education research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Number Page

ACKNOWLEDGMENTS

For the past several years, I have lived, worked, and researched on the lands of the Coast Salish peoples, land which touches the shared waters of all tribes and bands within the Suquamish, Tulalip and Muckleshoot nations. Check out `https://native-land.ca/` to find out which Indigenous people's lands you live and work on. If you're able, donate directly to the Indigenous nations and peoples around you affected by legacies of settler colonialism.

To my graduate advisor, Dr. Amy J. Ko—I sat here for nearly half an hour trying to figure out how to thank you for the years of support and guidance. You live out a model of scholarship that puts people first and a model of leadership that uplifts those around you. Thank you for everything. I hope one day I can run a research group with even half as much rigor, compassion, and poise as you exemplify every day.

Many thanks to my undergraduate advisor, Dr. Margaret Burnett, who introduced me to the world of research and taught me much of the the hidden curriculum of academia. If you had told the small-town kid who wandered into your office with vague questions about user interface design that they would be getting a Ph.D. in (almost exactly!) ten year's time, I would have laughed. But, you opened my eyes to the possibility that a person can do more with a computer science degree than just software development. Thanks for showing me how I could fit into computing research, teaching me how to rigorously explore the world, and inspiring me to reach for levels of attainment that I hadn't even considered a possibility before.

To my lab members, broadly construed, current and former: Good scholarship stems from community, and I couldn't have asked for a better one! Thanks for the research guidance, the collaborations, the philosophical musings, the emotional support baked goods, the laughs, the bad memes, the chance to be inspired by your awesome work and your awesome selves, and the innumerable baskets of fried potato products shared at many happy hours. You're all going to do amazing, important, impactful things and it'll be my absolute honor to cheer you on.

Thanks to the iSchool, DUB, and EduCHI communities for their ongoing support. I've had a blast organizing and participating in events alongside you all, and I hope I can carry these communities' infectious enthusiasm for making the world a better place to wherever I end up next. I've learned so much from so many people in these spaces that I won't even try to list them all here, but I thoroughly appreciate every bit of it. Here's to many more years of fruitful collaborations!

To my buddies in the Snohomish County APA pool league, thanks for being such a great community and reminding me not to take school and work too seriously. I felt the blue foam finger waving aggressively at me in spirit throughout it all.

The stories we tell ourselves in dark times are important, and the stories we tell each other even more so. Thanks to my excellent friends in my tabletop gaming groups for providing some light and levity when the world starts to look a bit drab. To the crew of the DonTreaders, the Spooky Game and Bleak House collectives, the Friends of Karl, and Her Majesty's Heroes—I look forward to saving the world(s) again with you soon!

Eternal gratitude to my wonderful partner, H. I may have been able to do this without you, but I certainly wouldn't have wanted to. Thanks for your love and support, especially in the final stretch—you kept me grounded.

And finally, to Rosie, my emotional support void and solid contender for best cat in the world. Here's to many more treats and ear skritches.

DEDICATION

To Calvin: As always, we carry on.

Chapter 1
INTRODUCTION

Given the prevalence of technology in today's connected world, the need for inclusively designed hardware and software is not only a matter of convenience and comfort—it is a matter of equitable and just access to information and resources. The harms of poorly designed technology disproportionately fall upon minoritized groups[1]. Prior work in human-computer interaction (HCI) and user experience (UX) design spaces documents a plethora of cases in which technology fails to support users with varied physical or mental capabilities [252, 279, 317], races or ethnicities [224, 245, 255], cultures [7], genders [57, 131, 209], and socioeconomic statuses [208], among other facets of diversity.

Of the myriad different types of design that go into creating a technological artifact, *interface* design can be a poignant site of inquiry for demonstrating the concrete impacts of design bias. Computing interfaces are the point at which technology designers' conceptions of the world meet the realities of users' everyday lives. Interfaces constrain allowable interactions with technology [306], and by extension, who gets to benefit (and who is harmed through exclusion) from technological access. As human-made artifacts, computing interfaces embed the values and biases of their creators [100, 101], often leading to technology designs that fail to meet the needs of users from different minoritized groups [69]. However, because technology carries perceptions of objectivity [18], it can be difficult to grasp how subjective design decisions made throughout the design process might result in more or less usable technology, especially among individuals with more technical backgrounds [230, 268].

To realize more equitable technology futures, it is not enough to simply adapt technology to be

[1]I use *minoritized* as a descriptor for groups who are disadvantaged by systemic injustices. Minoritized groups are the complement to *dominant* groups, who are unstigmatized [251], positively privileged [296], and generally favored within social, political, economic, and educational systems [89, 204].

more inclusive *after* it is created: We will also need to equip technology creators with the skills they need to critically reflect upon bias and exclusion *during* the technology design process [111]. In some cases, the people making computing interface design decisions are dedicated UX designers with explicit design training. However, in many cases, computing professionals (developers, software engineers, etc.) are in charge of interface design decisions, such as in small startups [174], companies lacking design cultures [174], open-source projects [176], and even larger companies where engineers manage and collaborate with designers [193].

Though computing professionals enter the discipline in many ways, formalized computing education can be one avenue to develop inclusive design skills in the students who will become tomorrow's technology creators and consumers [228]. Many factors influence whether computing professionals are able to enact justice-centered design values in practice, including organizational culture and access to resources [54, 295]. Since education impacts future practice [201, 203], helping computing students learn to critically reflect upon the ways their design decisions impact users can be one aspect of developing inclusive computing interface design competence. However, the most effective way to integrate inclusive design and computing education remains an open question. Many of the existing computing interface design methods created to promote inclusiveness (e.g. [24, 208, 217, 277]) are intended for use by professionals or people who otherwise have design experience. There are comparatively few resources and techniques for teaching inclusive computing interface design principles to students, much less students with computing backgrounds. My work seeks to address this gap, contributing pedagogical foundations to support computing students with little-to-no design background in learning to create inclusive computing interfaces.

1.1 Dissertation Outline and Research Studies

This dissertation explores how best to support computing students in learning to make inclusive computing interface design decisions, with a broad goal of equipping them with the skills to make inclusive technology in their future careers.

In Chapter 2, I provide background information on my pragmatic theoretical approach to this dissertation research. I draw upon prior work from HCI, software engineering, justice-centered design education, and computing education to situate my work. I conceptualize a justice-centered definition of inclusive design for use throughout this dissertation and describe how design decisions based on erroneous assumptions uphold normative structures of technological marginalization.

In Chapter 3, I describe a study involving qualitative analyses of existing computing curricula and standards through a design-informed lens. This investigation surfaced a dual-type theoretical model of the different kinds of design activity within computing education, noting that traditional computing education often neglects the teaching of design skills that help students situate technology in the broader world and understand its impacts.

In Chapter 4, I describe a study involving surveys and interviews with computing students and educators taking or teaching computing interface design courses which sought to uncover student learning challenges. This investigation revealed 18 challenges that prevent computing students from learning design skills, including several that could inhibit their ability to learn or practice inclusive design.

In Chapter 5, I describe the development of a theoretically-grounded design evaluation method to teach inclusive design skills called CIDER (which stands for Critique, Imagine, Design, Expand, Repeat). CIDER uses the critical lens of *assumptions about users* to reveal the ways normative design decisions make computing interfaces less accessible to users from minoritized groups. I also present a case study evaluation of CIDER's efficacy in an introductory interface design course, which found that the technique helped computing students identify increasingly more types of design bias over time, consider more diversity when designing, and adopt more inclusive design approaches in subsequent design work outside of the course.

In Chapter 6, I describe an Action Research study with a community of computing educators interested in integrating critical perspectives on interface design into their courses using the CIDER technique as a focal instructional method. This study revealed several instructional challenges that arose around the integration of critical topics in computing education contexts and several strategies educators tried to mitigate these challenges, contributing pedagogical content knowledge foundations for teaching computing students to make more inclusive design decisions.

Finally, in Chapter 7, I interpret the results of my studies in light of my broader goal of integrating inclusive design and computing education, including implications for research, practice, and future work. I conclude with a call to re-imagine the priorities of HCI and computing education toward a future where computing professionals are better equipped to critically reflect upon the individual and societal impacts of their interface design decisions.

1.2 Thesis Statement

This dissertation demonstrates the following thesis statement:

> Two forms of design decisions manifest within computing education: *program-space* design decisions that rely on technical implementation knowledge, and *problem-space* design decisions that rely on knowledge of the broader world. Computing students particularly struggle to learn problem-space design concepts, inhibiting their abilities to make inclusive design decisions. Explicit guidance on identifying the *assumptions about users* embedded in computing interfaces encourages students to make inclusive design decisions. When teaching inclusive computing interface design topics, computing instructors develop critical-specific pedagogical content knowledge (PCK) that differs from PCK for general computing instruction.

1.3 Contributions

My work makes four contributions.

First, I contribute a dual-type model of design activity present in computing education contexts, highlighting the need for further investigation into design decision making skills that help computing students understand the societal impacts of technology.

Next, I contribute a set of student learning challenges that arise in introductory interface design courses which may prevent computing students from developing the skills they need to design inclusive technology.

Then, I contribute a pedagogical technique that uses a novel strategy called assumption elicitation to help computing students learn to recognize and respond to computing interface design bias, as well as a case study evaluation of its efficacy and considerations for its use in post-secondary design learning contexts.

Finally, through an Action Research study of this technique's integration into post-secondary computing courses, I contribute pedagogical content knowledge for teaching inclusive interface design in computing courses. This includes descriptions of 11 instructional challenges that arose in their courses, each with a general instruction component and a critical-specific nuance, as well as 23 mitigation strategies they used to address these challenges.

These contributions extend existing work on justice-centered approaches to computing education and provide foundations for future work in the nascent sub-discipline of critical HCI education research.

1.4 Positionality

Before launching into descriptions of research on teaching others to consider different facets of human diversity in their work, I would be remiss not to reflect upon how my own identities and lived experiences impact my approach.

In all of this work, I humbly acknowledge the unique standpoint from which I conduct this research as well as the breadth of perspectives that I do not have access to. At the time of writing, I am a relatively young, white, able-bodied, English-speaking, neurodivergent queer person living in the western/global north culture of the United States Pacific Northwest. I have simultaneously experienced some kinds of design exclusion as the result of normative interface design biases (e.g. cis-heteronormative categorization schemes that force me to lie to proceed through forms [306]) while avoiding others (e.g. algorithmic profiling that discriminates against users from minoritized racial and ethnic backgrounds [18, 81]). I have had the immense privilege of attending well-regarded public universities and working with leading researchers in software engineering, human-computer interaction (HCI), and computing education. However, I was also raised in a relatively low socioeconomic status, working class rural community with little-to-no access to computing education and low overall levels of technological literacy.

Though these and other experiences drive me to conduct this research on making technology more inclusive and usable, they are also limited in scope. As mentioned throughout this dissertation, I recognize that one person alone cannot (and should not) speak to all the perspectives needed to truly advance equitable, critical, liberatory HCI education goals. We will need to conduct this work together, in all the brilliant messiness of a community that sees and values each other as full human beings, not simply as data points to strengthen arguments. I attempt to uphold this ideal in my empirical research by using largely qualitative methods that enable my participants to speak for themselves through rich, contextualized quotes, and, where possible, using community-based participatory methods like Action Research [319] to explore issues of importance to the people I work with.

Chapter 2
BACKGROUND & THEORETICAL FOUNDATIONS

In this chapter, I describe the background against which I situate my dissertation work. I begin by discussing the pragmatist epistemological approach that I work within to contribute applied solutions to real-world challenges that computing students and educators experience when trying to teach and learn inclusive design topics. I then conceptualize a justice-centered definition for *inclusive design*, drawing on design justice principles to discuss how it motivates my research on computing interface design decisions. Next, I provide an overview of literature on design rationale in the context of software and hardware design. I highlight how computing interface design decision-making relies on assumptions to make up for imperfect knowledge of design contexts, and how erroneous assumptions can lead to design bias and marginalization. Finally, I provide several working definitions of terms used throughout the dissertation.

Portions of this section have been previously published in my own prior work [228–231] and adapted to this format.

2.1 Theoretical Framing: Pragmatism as an Epistemological Approach to Solving Real-World Problems

Pragmatist epistemologies are situated, problem-focused, solution-oriented approaches to knowledge production and interpretation, the primary goal of which "to create practical knowledge that has utility for action for making purposeful difference in practice" ([116], quoted in [164]). Meaning within pragmatism is found in consequence [164]. Pragmatism posits that the "truth" of an idea or object exists inseparably from its impacts upon the world, and more importantly, from peoples' *experiences* and *perceptions* of those impacts [188].

Pragmatist epistemologies also account for pluralistic views of truth and impact: According to Oquist, "[t]here may be several sets of ideas that are equally plausible, good, and valid" under pragmatism, and "[t]he results of any operation are equally good if they contribute to the solution of the problematic situations that originated the inquiry" [232]. Worldviews under pragmatism can be simultaneously individually unique (since everyone's understanding of the world is influenced by their lived experience) and socially shared (because sets of similar experiences contribute to shared understandings) [164].

Research under this paradigm values mixed-method inquiry, using the approach that works best to answer research questions rather than adhering to quantitative or qualitative methods solely out of principle [164]. Several liberatory philosophies position themselves under pragmatism [188], and pragmatic research approaches can be particularly useful to tackle social justice issues [164]. Within HCI, pragmatism has been used as a lens to examine the discipline's history of UX evaluation goals [199], placed in conversation with rationalism to better understand design practice [110], and used to frame barriers to inclusive design work [166].

For my own work, I adhere to the pragmatic ideals of *value through impact* and the need for *situated interventions*. Biased technology negatively impacts the lives of real people every day, and since we will never live in a perfectly accessible or inclusive world, we need measures to mitigate these harms insofar as possible. The studies described in this dissertation are motivated by the pragmatist need to understand the existing landscape and barriers to teaching inclusive design effectively in computing contexts. To be effective, educational interventions must also be situated within the realities of today's world. Accordingly, similar to other efforts to advance equitable computing education [259], I opt to approach this work through the lens of *harm reduction* rather than framing it as a panacea for eliminating biased technology. My research contributions cannot erase design bias from the world, but they *can* help realize a future where fewer technologies are as egregiously biased in the first place by providing computing students with ways to concretely, critically reason about the impacts of what they create. Alongside other efforts to integrate critical perspectives into computing [178, 313], the insights from this body of work can contribute to a more equitable future of technological interaction for all.

2.2 Inclusive Design: Conceptualizing A Justice-Centered Definition

2.2.1 Design justice as a frame for inclusive design pedagogy

Some critiques of pragmatism posit that the hyperlocal, situated focus of pragmatic problem-solving can limit its ability to accurately identify and account for systemic, structural problems [164]. Toward this end, I also situate my work within *design justice* [69, 81] goals. Design justice frames design decision-making as both *abductive* (drawing best possible conclusions from incomplete information) and *speculative* (envisioning potential futures) [70].

Rooted in traditions of Black feminist thought including Collins' matrix of domination [63] and Crenshaw's intersectionality [71], design justice approaches actively recognize, attend to, and resist the ways hegemonic power imbalances manifest within design work. Costanza-Chock, in situating design justice within the broader world of design, notes that

> "universalist design principles and practices erase certain groups of people, specifically those who are intersectionally disadvantaged or multiply burdened under white supremacist heteropatriarchy, capitalism, and settler colonialism. What is more, when designers do consider inequality in design (and most professional design processes do not consider inequality at all), they nearly always employ a single-axis framework. Most design processes today therefore are structured in ways that make it impossible to see, engage with, account for, or attempt to remedy the unequal distribution of benefits and burdens that they reproduce." [70]

Design justice approaches are not limited to the realm of technology, but since computation amplifies power (and accordingly, any power imbalances embedded within it), software and hardware design processes are illustrative sites of design justice inquiry.

Design justice pedagogies are informed by elements of popular education [98], constructionism [2], participatory action design [82], and liberatory philosophies of teaching [143], with a heavy emphasis on non-exploitative participation of minoritized communities. Learning and teaching design justice requires that students "actively develop their own critical analysis of design, power, and liberation, in ways that connect with their own lived experience" and that educators "find methods to help students challenge their own ideas about themselves, their relationship to design partners, and the role of design in the world" [70].

2.2.2 Inclusive Design: Beyond "average" users

Many technological design approaches and frameworks emphasize the importance of designing for diverse kinds of users, such as universal design [42], ability-based design [307], value-sensitive design [100], and participatory design [213]. *Inclusive design* approaches aspire to make artifacts that are usable for as many people as possible, particularly emphasizing the needs of historically minoritized populations. This idea is popular in industry-focused design resources, such as those from Microsoft [210] and Adobe [5]. Some models of inclusive design operationalize inclusion as a proportion of a target population who can successfully use or benefit from a design as intended [167, 168], which includes not only ability-related concerns, but also characteristics like technology access, psychological factors, and access to support [121]. These kinds of metrics can help designers advocate for inclusion to clients and managers [318]. However, it can be difficult to authentically estimate the proportions of minoritized users within various populations, since typical classification structures encourage compliance to oppressive identity norms and often erase intersectional identities [25].

Most definitions of inclusive design also consider inclusiveness a situational variable rather than an innate property of a design. This is informed by the social model of disability [260], which holds that individual variation in ability does not make someone *disabled*; rather, it is the context where these variations are not taken into account that is *disabling*. Interactions between the user and a technology *in context* determine a design's inclusiveness [17]. Thus, it is the designer's responsibility to recognize (and design for) the fact that what is inclusive in one case may not be inclusive for everyone in all cases. Context-sensitive inclusive design approaches can support the development of more equitable technology [209, 224, 245].

A key goal in inclusive design processes is for designers to understand the perspectives of different user groups. Many of them label this process "empathizing" with users, though several recent works critique empathy-based approaches for their tendency to center designers' interpretations over community members' lived experiences [19, 20, 37]. Designers might use inclusion-focused versions of analytical methods like cognitive walkthroughs [45, 123, 208] and persona-based techniques [7, 24, 217] to better understand the needs of different users and create designs that work for them.

While some existing work investigates how analytical inclusive design evaluation methods might be used in class contexts [7, 228], most of the methods described above are intended for design professionals, not novices who may never have been exposed to design before. I argue that inclusive design should not be viewed as a practice best left to professionals, but instead as an integral part of introductory design education, no matter the domain. My work contributes pedagogical techniques focused on inclusive design skill development to address this gap.

My own conception of what it means to design *inclusively* is heavily influenced by design justice notions. Design bias and exclusion disproportionately impact users who are part of one or more minoritized groups. Designers who remain ignorant of this reality risk reinscribing oppression by basing their designs upon flawed normative assumptions about potential users' identities, capabilities, or contexts. The pedagogical inclusive design technique I contribute in later chapters exemplifies one method that can help students recognize the power they wield with their design decisions and begin to resist dominant design narratives. Ideally, after becoming more aware of design bias with these kinds of techniques and sensitizing themselves to interfaces as a site of power contest, computing students will be more ready to engage with minoritized communities directly in ways that minimize exploitative "parachuting" [70] and techno-solutionism [22].

Drawing upon the above literature, Section 2.4 contains the justice-centered working definition of inclusive design I use throughout this dissertation.

2.3 Design Rationale and the Role of Assumptions

When designers make design decisions, they consider multiple variables like their understandings of the design space, its constraints, established requirements, and user needs and preferences, among others. The reasoning behind these decisions is known as *design rationale*. Prior work from the area of software design suggests that documenting rationale can lead to higher quality final designs [41, 187] because it allows for designers to better account for artificial limitations they unintentionally place upon the design space [262]. Making design rationale explicit can also provide guidance for future design re-use efforts and concentrate organizational knowledge that might otherwise be diffuse or implicit in single, visible locations [184].

A single designer (or even a team of designers) can never have perfect information about the world, their users, or unanticipated interactions of either of these with their proposed design. To account for this missing knowledge, they necessarily rely on assumptions when making design decisions [189, 294]. Assumptions made during the design process are not always explicitly documented or even consciously recognized on the part of designers [34, 240]. As a result, these assumptions can be vectors for design bias, especially when designers assume certain things about potential users' capabilities, social or cultural contexts, or access to resources [101]. Unless particular traits or characteristics about users are specified, technology designers tend to fall back on designing for users of socially dominant or majority races, genders, ages, cultures, and/or classes—even if the designer themself is from a historically minoritized group [70]. Explicitly surfacing and documenting assumptions made during the design process provides one way to catch potentially harmful biases and, ideally, to minimize their impacts on design decisions [40]. This notion is supported by prior work on the role of assumptions within software rationale documentation, which suggests that augmenting rationale with explicit assumptions can help identify intervention points for improving a design, because it enables detection of parts of a system that rest on incorrect assumptions [41].

If assumptions are not caught during the early stages of the design process, they might also be identified and addressed using various design evaluation methods. Empirical evaluation methods like usability studies, technology probes [149], or experience sampling techniques [65] can help designers identify implicit assumptions by making visible the ways that the design breaks down during use. Analytical evaluation methods like claims analysis [49], heuristic evaluations [220], empathy maps [155,267] and cognitive walkthroughs [298] can also help reveal erroneous assumptions designers made about a user's prior knowledge or preferred interaction styles. However, these methods are generally intended for use by professional designers. They rarely provide the scaffolding learners need to tie assumptions, design bias, and inclusion together. My work seeks to address this gap, contributing pedagogical foundations to support computing students with little-to-no design background in learning to recognize and respond to assumptions embedded in existing computing interfaces.

2.4 Working Definitions

Informed by the above bodies of literature, my working definition of *computing interface design* in this thesis draws on Park and McKilligan's model [233] and includes design practices related to interface, interaction, and user experience design for technological artifacts. I use the term *technological artifact* (or sometimes just *artifact*) inclusively to refer to both software and hardware with computational or computing-related components. For the purposes of this thesis, I define an *assumption* embedded within a design to be a way in which a design's features and affordances rely on a user to have particular capabilities, resources, means, or knowledge, without which they might find it difficult to interact with a technological artifact[1]. I then define *design bias* to be the ways in which assumptions might make it disproportionately difficult for particular users, especially those from minoritized groups, to interact as intended with a technological artifact. Finally, I define *inclusive design* broadly to be an approach to design work that recognizes the normative structures that may lead to design bias and attempts to account for them by intentionally designing artifacts to support the needs of minoritized users.

[1]For example, some assumptions embedded in the design of a standard US QWERTY desktop computer keyboard are that the user has enough fine motor control to press small keys in a particular sequence, or that they can recognize Latin/Roman alphabet characters.

Chapter 3
ON THE ROLE OF DESIGN IN K-12 COMPUTING EDUCATION

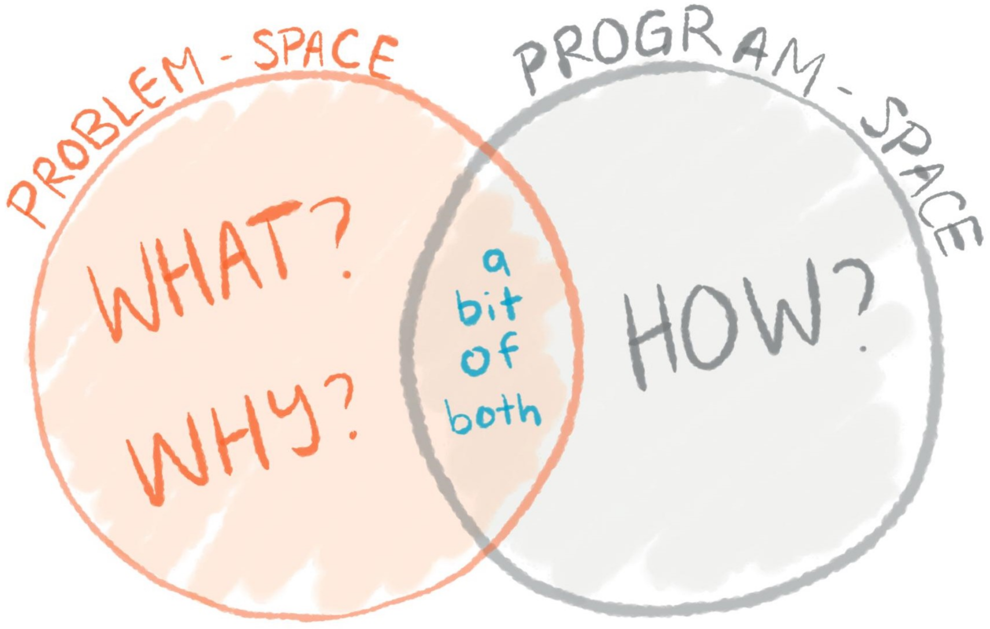


Figure 3.1: The study described this chapter found two distinct, yet overlapping kinds of design activity in computing education materials: problem-space and program-space design.

3.1 Introduction

The field of design is a distinct discipline, with entire academic departments, areas of scholarship, tools, practices, and professions dedicated to it. However, aspects of design often overlap in meaningful ways with other disciplines, blurring the boundaries between it and the fields in which its skills are used. Foundational design literature claims skills such as planning [8, 249], iterative problem-solving [6, 239, 250], and evaluation of an artifact's effectiveness, utility, costs, and values [151, 158, 218, 308] as core to the field[1]. Designers are clearly not the only ones who practice these skills, though. In disciplines like engineering and architecture, practitioners might create, evaluate, and execute project plans, carrying out design work while not necessarily identifying as designers themselves. This entanglement manifests in discipline-specific design subfields (e.g., engineering design, architecture design) which are neither wholly design nor wholly the intersecting discipline.

In professional practice, computing-related areas also experience this disciplinary overlap with design. Work from software engineering shows that developers often make design decisions as they create software. Developers and software engineers may find themselves solely responsible for user interface design in smaller startups or in companies that lack design culture [174], or they may act as gatekeepers for design decisions in open source projects [176]. Even in larger companies with in-house design teams, software engineers commonly collaborate with or even manage designers, often contributing to major design decisions [193].

We see evidence of computing's overlap with design in computing education contexts as well, often encoded into the definitions and standards of what it means to "know computing". The ACM 2005 Computing Curriculum documentation detailing a set of standards for postsecondary computer science education answers "What is Computing?" with the statement [281]:

> "[W]e can define computing to mean any goal-oriented activity requiring, benefiting from, or creating computers. Thus, computing includes *designing* and building hardware and software systems for a wide range of purposes;..." (emphasis added)

Similarly, standards and curricula for primary and secondary computing education often integrate design, often as major foci for learning activities. For instance, many of Code.org's *Hour of Code* modules, intended to be students' first exposures to programming, revolve around both envisioning (designing) and creating (engineering) a small animation or game. The first of the AP CS Principles' Big Ideas of Computer Science centers on *creativity* [10], which is often claimed as a central

[1]A more detailed discussion of what design is and what skills it includes can be found in Section 3.2.

design trait [6,80,84,154]. Efforts to broaden participation in computing per the computing education research (CER) goals established in 2014 [68] often rely on problem-solving [76,79,236,280] or creativity [97,104,142,248] to engage students, both of which involve design-related skills and practices. Influential perspectives on computing education may even refer explicitly to design as a means by which learners are motivated to engage with computing in the first place [127]:

> "A student who takes an introductory computer science course wants to *make* something. Even if the student doesn't want to become a professional software developer, they want to *create* software, to *design* something digital" (Guzdial, emphasis preserved)

Design seems to be intrinsically embedded into computing education contexts, especially at the primary and secondary levels.

Unfortunately, the entanglement of design and computing education often results in difficulties teaching, learning, and applying computing knowledge. Prior work around the teaching and learning of design within computing contexts (generally in human-computer interaction classes) repeatedly finds challenges engaging and promoting students' learning [91, 144, 206, 229, 247], assessing students' design-related competencies [31, 282, 316], and enabling instructors to teach design-related material effectively [55, 147, 228]. Further, design problems appear to pose particular challenges to those who create software interfaces [237], architectures [262], and requirements [1] once they enter the workforce, suggesting that whatever design-related education students receive is not yet properly preparing them to practice these aspects of computing professions. And yet, we can't simply ignore design's role in computing practice—to do so would misrepresent the field and, since prior work shows that design proficiencies can contribute to more pronounced computing expertise [202, 238, 261], runs the risk of being detrimental to computing students' career-readiness.

I claim that a number of these pedagogical challenges stem at least in part from a lack of understanding on how to characterize and effectively teach design-related skills and topics within computing education contexts. Recently, Wilcox et al. noted this deficit, contributing initial insights into the role of design in human-computer interaction (HCI) courses and calling for more research into the intersection of design and computing education [302]. Additionally, though myriad CER-related work analyzes computing curricula and standards through lenses of diversity (e.g. [36,42]), career choices (e.g. [51,314]), and culture (e.g. [15,102]), little to no work exists that analyzes computing education materials from a design lens, especially in the rapidly growing area of primary and secondary computing education. Postsecondary computing education materials and curricula are often specific to the class instructor or university, lacking the consistency and reach of materials like the CSTA standards or the Code.org or Exploring CS curricula. Further, there is growing administrative and political interest in integrating computing proficiencies into K-12 education. Given the recent and continuing uptake of computing education in K-12 settings, investigating this demographic in particular provides an opportunity to gain insights relevant to a much larger population than those who enroll in computing higher education.

Prior attempts to critically investigate and characterize discipline-specific manifestations of design at all levels have been shown to produce positive results. Design-centered analyses from K-12 and postsecondary engineering education, for instance, surfaced valuable insights around pedagogy and practice [73, 219, 275], and while theories of disciplinary applied design can be difficult to create and generalize, they benefit both practitioners and learners [74,114,269]. We can infer that computing education might find similarly informative results through investigation of the field's interactions and overlap with design. Gaining clarity about the nature of design in computing education contexts would likely help inform curricula and pedagogy about what it should mean to "know computing", especially in K-12 contexts where design is used to motivate learners. More clarity might also shed light on how to frame both design and computing to learners in a concise and understandable way.

This chapter provides an initial investigation into the nature of discipline-specific computing design in educational contexts, toward the goal of motivating and laying groundwork for further research into the overlap of design and computing education. To enable shared understanding and a working definition of design, I first draw upon literature from the design field itself and synthesize five basic cross-cutting skills of design work, which later form the basis of my analysis in

Sections 3.4 and 3.5. I then review related work from selected design-based education research sub-disciplines, with a particular focus on how characterizing discipline-specific manifestations of design supports improved teaching and learning outcomes. I present two exploratory, qualitative studies of popular K-12 computing education standards and learning activities. The first study asks *What is the nature of design and computing's overlap in K-12 education?* and finds evidence to suggest two major types of discipline-specific design appearing within computing education: problem-space and program-space design, which may overlap. The second study, informed by the results of the first, asks *How do problem- and program-space design manifest in K-12 computing education activities?* and finds evidence to suggest that these two types of design can exist separately, but that they often overlap in educational contexts, creating an intriguing intersection of design and computing activity. Finally, I discuss implications for teaching and learning computing topics that arise in light of these results, highlighting the need for further research in this area to support more effective computing education in both K-12 and higher education. [2] [3]

3.2 Background: What is Design?

3.2.1 Perspectives on Design

Definitions of design abound, resulting in many perspectives on what design is and what it entails. The goal of this section is to provide a functional understanding of the nature of design as well as to identify core skills of design as suggested by design literature.

[2]Throughout this chapter, I use first-person language ("I", "my", etc.) to describe this work to reflect the single-author nature of the dissertation document. However, this work was conducted in collaboration with Brett Wortzman and Dr. Amy J. Ko, and published in ACM Transactions on Computing Education (TOCE) Volume 21, Issue 1 in December 2020 [231]. I acknowledge the shared contributions of all authors to the original publication.

[3]We thank the third analyst, Aubree Ball, whose perspectives on design helped us better understand the discipline of design and how it interacts with computing. This material is based upon work supported by the National Science Foundation under Grant No. 1735123, 1539179, 1703304, 1836813, and DGE-1762114, and unrestricted gifts from Microsoft, Adobe, and Google.

Design as planning

Some early perspectives on design viewed it as a means to plan how an artifact fits into the context of the world. Design from this perspective is a single stage in the process of creating an artifact that "terminates with a commitment to a plan which is meant to be carried out" [249]. Designers analyze the problem space of a given issue, define concrete requirements for a solution's form, and propose a particular solution that fits the given constraints. This view has its roots in the systematic design movement [8, 38, 158]. Extreme versions of this perspective hold that any act of planning an approach, rather than tinkering and learning through trial and error, is an act of design [249].

Design as iteration

In contrast to design-as-planning's single-phase view, many hold that design is an iterative, reflexive process of finding the most fit solution to a given problem. Fundamentally, this perspective sees designing as "a process of error-reduction" [6]. Designers adjust a solution's form when they discover a mismatch between the it and the world's context, iterating through multiple planning, implementation, and testing phases to discover these mismatches. Central to this view are the ideas of productive failure [239] and modularization [6]. Many models of design processes incorporate iteration [86].

Design as expression

Some perspectives on design hold that the designer embodies aspects of themselves or their values within designed artifacts. For instance, design may be a means of communication [170, 218] or rhetoric [43, 83, 90, 205] (e.g., how Facebook's design expresses Mark Zuckerberg's view of a "post-privacy" world [157]). Each time a designer creates an artifact, they implicitly communicate about its nature in choosing how to represent it [218] and uphold particular values, whether they intend to or not [100,249]. Designed objects from this perspective are an extension of the designer's self and an outlet to express their creativity and emotions [30, 100, 170, 222].

Design as a means to address complex problems

Finally, many view design as the only way to address "wicked" problems [39]–ill-structured, often incomplete, complex issues not easily solved through traditional methods [250, 270]. Optimal solutions cannot be found for wicked problems unless unrealistic constraints are imposed, in which case the solutions fail to fully address the original context. Many of the most pressing sociocultural problems of today are wicked. Consider, for instance, finding effective strategies to address climate change or embedded social injustice, which are multifaceted and involve complex tradeoffs. This design perspective manifests itself in DesignX [223] and Transition Design [152], as well as in the popularity of "design thinking" as taught at Stanford's d.school [46].

3.2.2 Core Design Skills

In order to operationalize these somewhat competing perspectives in a concrete manner for my analysis in Sections 3.4 and 3.5, I found it useful to draw out the skills which foundational design literature claims as integral to design. Five skills in particular seemed to cut across the definitions of design discussed above, suggesting they are central to designerly thinking and practice. I note here that though this list of skills is the result of my own synthesis of the literature, it bears similarity to sets of design skills presented in other studies of discipline-specific design education research (e.g. [73]), implying a base level of validity.

Understanding the Context (UtC)

Designers use this skill to inform ideation. To fully understand the context, designers must learn about both existing systems that attempt to address a problem and any stakeholders who will interact with the designed artifact. Techniques employed while trying to understand the context involve primary or secondary user research or focus groups [86, 297] and often also involves perspective-taking or empathizing with target users [133, 274, 310]. From these activities, designers identify pain points and opportunities for improvement that exist in current systems. The information designers gain from this work forms the underlying rationale for design requirements. Gaining a rich understanding of the context is critical to ensure that designers adequately address their client's, customer's, or users' needs.

Creative Ideation (CI)

Creativity is central to design [6, 80, 84, 154]. Designers employ creative ideation as a skill when they begin to generate many possible solutions. The goal of this practice is not to fixate on a single idea, but to explore the solution space and generate many promising approaches. Creative ideation includes activities like prototyping (i.e. ideating through making) and otherwise externalizing one's ideas through sketching or verbal communication. This skill also encompasses practices like "stealing" successful ideas to analogous problems [118,134] and composing features of those ideas to create novel things [216, 289].

Evaluation & Synthesis (ES)

As designers generate ideas, they evaluate them against the constraints of the problem context. The goal of evaluation and synthesis in design is to practice convergent thinking and begin narrowing down the solution space to a handful of feasible options. Formal evaluation processes might include critique sessions [151, 308] to gain constructive feedback from others, evaluation against design heuristics [220], game testing or other empirical evaluations (e.g. [180, 299]), or code reviews. Informal evaluations also occur in the design process: Designers make their own expert judgments about which ideas are worth pursuing and feasible to implement. Designers often compare potential solutions against the time and resources they have available or the prioritization of certain user needs over others, performing an analysis of tradeoffs inherent in implementing one solution over another.

Iterative Improvement (II)

Design is an inherently iterative discipline. Designers use iterative improvement to optimize their solutions and incrementally update them to fit the problem space. This skill involves the ability to adapt solutions based on feedback, whether that feedback is from a teammate, a user, or even a technological system such as a debugger. Iterative improvement also necessitates an approach to failure as a learning experience, not as a catastrophic ending: Breakdowns in the design reveal opportunities for improvement [239]. Each iteration on a solution informs the designer about the problem space's constraints–information which is taken into account in successive iterations.

Communication (Comm)

Underlying and supporting all designerly practice is the skill of communication, especially with those in fields other than one's own. Though this skill is by no means exclusive to design, communication is a basic design competency [9]. Without the ability to communicate ideas to teammates, stakeholders, and the general public, designers cannot ensure that their designs are implemented correctly or that stakeholders understand their roles and responsibilities. Examples of this skill in practice include describing concepts and ideas in terms that a particular audience understands, presenting results in fitting manners, and working with teams (e.g., of developers, engineers, or stakeholders) to define realistic common goals [3]. Communication also encompasses the notion of field literacy: Designers must be knowledgeable enough in their stakeholders' fields and environments that they can interpret feedback and design a solution appropriate to the context.

3.3 Related Work: Applied Design Education

Though the core design skills described above remain relatively stable, they manifest in different ways when design overlaps with other disciplines. Many fields have benefited from rigorous investigation of the role design plays within them. Often, the knowledge gained from this work manifests in a subarea of the discipline's existing education research. For instance, there exists engineering education research, which studies the learning and teaching of engineering, but also engineering *design* education research, which studies the learning and teaching of engineering-specific (i.e. disciplinary) design. Here I briefly describe work from architecture design education and engineering design education, two disciplines in which an understanding of how design intersects with and manifests within in the field helped improve curricula and pedagogy. I also present relevant work from STEM education, highlighting how design activities are often used in classrooms to promote understanding of STEM concepts. Finally, I present related work on teaching design skills in computing contexts, framing this chapter's initial attempts to gain insight into the role of design in computing education as a way to ground future work in disciplinary computing design education research.

3.3.1 In Architecture

Architecture was one of the earliest fields to recognize and study its overlap with design in educational contexts. For instance, Schön's influential theory of *reflection-in-action* [257] arose in part from observation of architectural design studios. Schön used this theory about the nature of design in educational contexts to inform frameworks for design knowledge and pedagogical strategies to help novice designers become effective professionals [258]. Lawson studied how first-year and final-year architecture students acquired design skills, comparing both groups' cognitive strategies for problem-solving to scientists and non-designers. The results of this study not only improved pedagogy by characterizing gaps between first- and final-year design students, but also served as a basis for modifying CAD modeling tools to better enable architectural design practice [186]. Goel and Pirolli's recognition of the distinction between design and non-design (engineering) tasks arose from studying architecture students' problem conceptions, which enabled more tailored pedagogy for architecture design problem-solving instruction [114]. Notably, Goel and Pirolli also reported that the skills of design practiced by architecture students seemed to differ from the kinds of design practiced by mechanical engineering or curriculum design students, indicating a basis for discipline-specific design education research.

More recently, work in the area has built on these foundations to explore how well the current state of architecture design education prepares students to deal with the complex problems that will face them when they enter the workforce. Goldschmidt and Sever discovered differences in the role of creative ideation in undergraduate and graduate designers' processes: Undergraduate architectural and industrial designers tended toward creating products, while graduate students tended to create services or systems [117]. Based on this analysis, Goldschmidt and Sever critiqued the effectiveness of architecture design pedagogy, suggesting that educators focus on preparing students to address ill-structured "wicked" problems rather than teaching design methods. Chiardia et al. proposed educational strategies for integrating values in urban design education (an offshoot of architecture), instilling ethics and value-sensitive judgement-making into student designers [52]. These and other related works contributed to discourse around the nature of design in architecture education, affording both pedagogical improvements and further refinement of the discipline's conceptions how design manifests these kinds of contexts.

3.3.2 In Engineering

Engineering design education, while academically younger than architecture design education, has similarly worked to identify the role of design within its discipline. Engineering design is similar to computing design in that the two disciplines involved appear to be distinct on the surface, even though engineers often make design decisions in practice. Sim and Duffy's early attempt to categorize the ways design intersected with engineering education resulted in a generic ontology of engineering design activities [269], which shed light not only on classroom activities, but went on to inform product life cycle management [173] and system design [113] as well. Smith et al.'s work on characterizing "pedagogies of engagement" in the engineering classroom found that while while engineering students tended to be more engaged in project-based learning classes where they both designed and implemented their ideas, there were still many open pedagogical and curricular questions about how exactly to teach engineering design practices [273]. Informed by this work, Prince and Felder found that pedagogies which implement engineering design well were often more effective at promoting learning than traditional engineering education [242].

A notable body of work in engineering design education focuses on the development of design expertise in engineering students. For instance, Lemons et al. studied how physically building models of design artifacts interact with engineering students' design expertise [191]. Though the authors explicitly caution against interpreting their results too broadly due to the limitations of their sample, they found evidence to suggest that model construction can enhance creative thinking and enable awareness of metacognitive strategies – both of which may contribute to design expertise development. Atman et al.'s series of studies on the differences between freshman and senior engineering design students (e.g. [11, 12]) helped define concrete learning goals for engineering design pedagogy. To augment this applied work with theoretical understanding, they later analyzed nearly a decade's worth of this work through the lens of Schön's reflective practitioner theory [257], concluding that certain observable classroom design behaviors are reasonably indicative of design expertise [4]. Leveraging the insights afforded by this investigation, they identified problem-setting and engaging in reflective conversations with the design space as two trends of interest in design expertise representation and suggest that future engineering design education research should explore effective ways of imparting these skills to students.

Within the last decade, multiple researchers have called for more emphasis on discipline-specific engineering education research (e.g. [103, 272]), spurred in part by the trend toward discipline-based educational research, but also by the promising results already emerging from the field. Crismond et al. established a comprehensive framework of pedagogy needed to teach K-12 engineering design effectively, encompassing student misconceptions, teaching strategies, and more [73], helping to define the role design should have in engineering education. Starkey et al. investigated how engineering students' creative ideation processes change throughout the course of a project, finding that the design task itself has an impact on creativity and laying the groundwork for further educational research into how to teach creativity to engineering students [275]. Some of the newest work in engineering design education, Neroni and Crilly's exploration of how engineering students experience design fixation in isolation and in groups, both added to the general body of work on design fixation and suggested engineering-specific pedagogy for decreasing fixation [219]. These accomplishments around the learning and teaching of design in engineering contexts would almost certainly not have been possible without the work which came before it which helped clarify the role of design in engineering.

3.3.3 In STEM Education

In STEM education, design activities are often used not to impart design skills themselves, but to help students learn about STEM-related topics through designing. For instance, Hmelo-Silver's Problem-Based Learning (PBL) approach situates students as active participants in their learning processes [140]. Many of the goals of PBL, such as the development of problem-solving and collaboration skills in students, overlap with the goals of design education and the crosscutting skills of design discussed in Section 3.2.2. Hmelo-Silver et al. contributed a detailed study of how a Learning By Design (LBD) approach [181, 182] could help middle school students gain knowledge of the human respiratory system, finding that students in the LBD classes showed evidence of better learning outcomes than those who received traditional instruction [139]. They gave several recommendations for educational practice when including design activities in instruction, but noted the importance of having sufficient time for instruction in this style. Later, Hmelo-Silver and Pfeffer studied the differences in how experts and novices think about complex systems (in this case, the ecosystem of an aquarium) and found that while novices organize knowledge based on perceptually salient structural features (e.g. fish, plants, filters), experts organize knowledge at the behavioral (what the filter does) and functional (why the filter is necessary) levels [141]. This is consistent with prior work in physics education on differences between novices and experts [29]. This line of work later informed the pedagogical strategy of having students describe complex systems in terms in the form of structure-behavior-function models, which promoted deeper understandings of the system as a whole [291].

In a similar fashion, a substantial body of work on engaging students in designing hypermedia and multimedia systems (e.g. [50, 179, 197]) indicates that engaging primary and secondary students in learning through design activities can support higher-order cognitive skills needed to effectively contribute to complex projects. Lehrer offers an example of how secondary American history students took on a more active role in their education when given the opportunity to design systems that reflected their knowledge [190]. Liu and Pedersen studied how hypermedia authoring impacted primary students' higher-order thinking and found that the kinds of skills practiced during design activities contributed to students' holistic growth as learners [196]. In particular, they note situating skills such as planning, critical reflection, and collaboration within design activities may help students learn to value these skills more highly, which may contribute to success in later project-based education.

Further, design activities in STEM education can promote student engagement and create a more inclusive classroom culture. Kafai's line of work on constructionist learning through designing games [160] aims to help students engage more deeply with STEM concepts. Early on, Kafai et al. explored using game design activities to promote mathematical learning and found that both students and educators who engaged in design practices created higher quality games, thought in more principled ways about the topic of instruction, and better leveraged their real-world knowledge bases to understand mathematical concepts [162]. They later framed constructionist educational game development as a way of enabling early ownership and participation in digital culture [159]. Others have highlighted the potential for equitable education which arises from design-based instruction. After implementing their Learning By Design (LBD) approach in middle school science classrooms, Kolodner et al. found that LBD activities helped build a classroom culture in which students of diverse backgrounds and motivations were more comfortable engaging with scientific concepts, often showing evidence of improved learning outcomes compared to control classes [181].

Overall, much of the work concerning design in STEM education uses design as a means to enable learning of other, non-design-related concepts. While this literature presents many valuable insights into how educators can leverage design for use in their classes, using design as a tool to promote understandings of STEM topics is not the same as investigating the teaching and learning of embedded disciplinary design skills. Compared to architecture or engineering design education, relatively little prior work in STEM education positions its main goals as attempts to establish how design manifests in discipline-specific design education (e.g., studying the forms of design which exist in primary and secondary math or science classrooms). In this chapter, my goal is to understand the nature of design *itself* in computing education, not necessarily how design activities can be used to better impart computing proficiencies.

3.3.4 In Computing

Computing education research is a young discipline. As a result, we do not yet have an established subarea for disciplinary design education research (analogous to architecture or engineering design education research). Software engineering research provides some insights into the nature of design's role in computing practice. For instance, software professionals often encounter design-related challenges in practices such as requirements elicitation [1] and interface creation [237]. These challenges may be compounded by the often blurry boundaries between the software design and software implementation. Unlike more traditional fields in which designers work with physical media like wood or metal, software designers work with code and are not limited by the constraints of physical material (load bearing capacities, conductivity, etc.) [237]. These unclear distinctions lead to software development processes in which activities of defining and implementing requirements are tightly, iteratively coupled, such as Boehm's spiral model of software development [23].

A small but rapidly growing focus on education exists in human-computer interaction (HCI) literature, studying in part how computing students learn the design skills needed to create usable, useful software interfaces. This body of work has identified many practical challenges of teaching and learning HCI. For instance, it is often difficult to reliably engage students in HCI classes [144, 206, 247], especially when they view the course content as "inessential" [55], "easy", or "commonsense" [91]. HCI educators also find it difficult to assess students' design work [31, 282, 316], perhaps due to a lack of pedagogical content knowledge for design-related HCI topics [228, 229]. Overall, however, Lewthwaite and Sloan found that there is no agreed-upon pedagogy for teaching HCI principles and that most HCI education work "comprise[s] of teachers' reflections on their own practice and course design" [192], making it difficult to synthesize a generalizable notion of design in HCI from existing literature. Given that HCI education faces a time crunch in already overcrowded computing curricula [55, 130], educators must prioritize some topics and exclude others, but there is no agreement on what the core topics of HCI even are [122].

Though software engineering work describes many challenges developers face that involve disciplinary design, it does not necessarily provide the clarity required for computing educators to teach software design skills effectively or to know how students will respond when learning them. Similarly, HCI education work excels at identifying challenges that exist in design-related computing education, but it also appears to lack unified notions of around the role of design in computing education contexts. The goal of this chapter is to help establish this much needed clarity so that educators and researchers can address these challenges and create more precise, useful, and effective curriculum and pedagogy. Recent work from Wilcox et al. noted the lack of existing insights into the nature of design in higher education HCI instruction and called for further educational research to untangle the nature of design in supporting computing learning [302]. I hope to build upon this work by providing additional insights into the role of design in K-12 computing education, which has not yet been analyzed through a design lens, and proposing a conceptual taxonomy of design in computing that may be useful at all levels of instruction.

3.4 Study 1: What is the nature of design and computing's overlap in K-12 Education?

To begin building an understanding of how design skills manifest in computing education contexts, I performed a deductive qualitative analysis [235] on three sets of popular K-12 computing curricula and standards. I examined the learning objectives from each curriculum for the presence of the five core design skills from the literature described in Section 3.2.2. I chose to analyze the curricula and standards at the learning objective level because learning objectives and skills represent similar granularities: achieved learning objectives imply proficiency in certain skills. Computing education learning objectives that imply proficiency in design-related skills may reveal connections between design and computing and help us to understand the nature of the two disciplines' intersection.

3.4.1 Sampling Rationale: Typical Cases

To select the sets of learning objectives, I employed Patton's group characteristics (typical cases) sampling strategy [235] to identify curricula and standards that might be representative of many K-12 students' computing education. This resulted in three sets of learning objectives from the following sources[4]:

- The *CSTA K-12 CS Standards* [64] list what competencies primary and secondary students should have to claim computing proficiency. Most recently updated in 2017, these standards have informed many introductory computing curricula in multiple countries [88, 234].

- Code.org's *CS Discoveries* curriculum takes a self-proclaimed "wide lens on [CS] by covering topics such as programming, physical computing, HTML/CSS, and data" [59]. Freely available online, these materials are designed for early secondary education.

- The *AP CS Principles* course for late secondary education helps students gain familiarity in basic computing concepts, regardless of whether they intend to major in computing in post-secondary education. A record 70,000 students took the AP CS Principles end-of-course exam in May 2018 [61].

Many nations engage in discourse around the boundaries and core concepts of computing education. For instance, the United Kingdom, Australia, and Korea have all participated in the development of primary and secondary computing education standards and curricula, along with numerous other countries. Since it was not feasible to study all the primary and secondary computing curricula that exist, I necessarily had to scope my analysis. For this initial study, I chose three sets of learning objectives developed primarily for U.S.-centric learning contexts, which may limit the generalizability of the findings. Nonetheless, these learning objectives are representative of many students' computing educations. A 2019 report on the status of computer science education policy in the U.S. indicated that 39 of the 50 states had adopted CS standards for education [60]. The standards states adopted were almost always informed by the CSTA standards. As mentioned before, tens of thousands of students take the AP CS Principles exam each year, and more than

[4]Each of the three sets of learning objectives for Study 1 were collected on January 24, 2019 from the sources found in the bibliography. My analysis reflects the published materials at that time, and does not cover updates between the time of analysis and the time of publication (such as Code.org's recent rework of the CS Discoveries Design Process unit).

100,000 teachers have participated in Code.org's CS Discoveries professional development program to date. Because of this, I feel that the three chosen sets of objectives are sufficient for this exploratory investigation. Future work should apply this mode of analysis to other countries' objectives and standards in order to deepen our understanding around the role of design in computing education in global contexts.

3.4.2 Analysis of Learning Objectives

To ensure that my analysis encompassed as many diverse, relevant perspectives as possible, I had three individuals with varying backgrounds analyze the learning objectives for the presence of design skills. All analysts were either researchers or educators at a large, public, U.S.-based university. They were:

- The dissertation author, a computing education researcher with five years of research experience in HCI and design methods at the time of analysis, including two years researching the overlap of design and computing education.

- A CS educator with nine years of experience teaching secondary and post-secondary CS in the U.S. and significant experience designing primary and secondary CS curricula and mapping them to standards. After performing their analysis, the CS educator became interested in the project and joined the research team, becoming the second author on the eventual publication.

- A design educator with four years of experience teaching design (including interaction design) in the U.S. at the post-secondary level and seven years of experience as a practicing industrial and architectural designer.

The analysts' diverse backgrounds are essential to understanding the nature of the intersection of design and computing. To achieve robust and reliable understanding, it is necessary to (attempt to) surface all possible connections between design and computing from as many relevant perspectives as possible. This goal would not be possible if the analysis comprised of three similarly trained individuals identifying connections: They would likely miss important relationships between the two fields due to their limited scope of expertise. Correspondingly, my goal for this deductive analysis was not to reach inter-rater agreement (a metric often used to evaluate the consistency of qualitative judgments). Instead, consistent with the view of qualitative work presented by Hammer and Berland [132], I hoped to gain a holistic view of the role of design within computing from many relevant perspectives and surface multiple diverse interpretations.

Each analyst performed their mapping independently. I provided analysts with detailed descriptions of each design skill similar to those presented in Section 3.2.2. For each computing learning objective, I asked analysts to determine which (if any) design skills they felt were represented in the objective, allowing for multiple skills to be represented in a single objective. I also encouraged analysts to apply their individual expertise when determining which learning objectives mapped to each skill. After each analyst completed their work, the first two analysts collaboratively examined the results for patterns.

3.4.3 Results

Design Skills in Computing Learning Objectives

In total, the analysts examined 384 learning objectives for the presence of core design skills.

As intended, the analysts surfaced diverse interpretations of which computing learning objectives contained design components. For example, the CS educator interpreted any objective that involved creating a digital artifact or developing a computer program as aligning to the *Creative Ideation*, *Evaluation & Synthesis*, and *Iterative Improvement* skills, as, in their experience, these skills are inherently exercised in the process developing a program or digital artifact. The computing education researcher took a more conservative stance due to their prior knowledge of both design and computing, leading to fewer overall identified instances of design in their results. The design educator identified some learning objectives as falling under a sixth skill they named *Analysis and Synthesis of Information*. After the design educator discussed the rationale behind their choice with the other two, the analysts slightly revised the descriptions of the original five core design skills to clarify language around analysis and synthesis (though they elected to retain only the five existing design skills from the literature) and audited their analyses against the new descriptions. The results presented below reflect the audited analyses.

Some learning objectives did not seem to imply any of the five core design skills, indicated by a lack of design skill codes by any analyst. As might be expected, these were often the most obviously computing-focused objectives, encompassing skills typically ascribed to traditional computer science. Many of these objectives revolved around basic familiarity with computing concepts, such as *"Define an algorithm as the series of commands a computer uses to process information"* (Code.org 1.6) or *"Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data."* (CSTA 1A-DA-05). Other learning objectives that did not seem to contain design skills were more engineering focused, often simply asking students to implement programming constructs, such as *"Create and link to an external style sheet"* (Code.org 2.10), *"Create procedures with parameters to organize code and make it easier to reuse"* (CSTA 2-AP-14), or *"Create and modify an array"* (Code.org 6.10).

The classification of many learning objectives was more ambiguous. Two modes of conflicting agreement surfaced in the objective analyses, delimited by how they manifested in the coding results. (These two types, by their definitions, comprise the entirety of analysts' disagreement.) One kind of conflict occurred when some analysts identified design in an objective, though others did not. The most common form of this was the design educator and the computing educator agreeing that an objective contained particular design skills while the computing education researcher did not. In particular, the two educators took a broader stance than the researcher on what kinds of learning goals implied *Understanding the Context* and *Communication* skills. For instance, both educators saw UtC and Comm in objectives like *"Explain the beneficial and harmful effects that intellectual property laws can have on innovation"* (CSTA 3A-IC-28) and *"Explain characteristics of the Internet and the systems built on it"* (AP CS P 6.2.1). The computing education researcher did not see design skills in these objectives, conservatively interpreting them as asking students to relay information they had read or heard. A second type of conflict occurred when all analysts indicated there was some kind of design within an objective, but disagreed on which particular design skill(s) it contained. For example, each analyst mapped *"Develop a correct program to solve problems"* (AP CS P 5.1.2) to three design skills. The design educator mapped the objective to *UtC*, *CI*, and *ES*; the computing educator to *CI*, *ES*, and *II*; and the computing education researcher to *UtC*, *II*, and *Comm*. This learning objective's description contained examples of what students might do to achieve the objective, though it appears that the analysts all interpreted these examples differently and focused on different aspects. Similar patterns occurred throughout the analysts' results in all three curricula. The two kinds of disagreement surfaced by the analysis highlight the importance of including many diverse perspectives in my initial attempts to understand the nature of discipline-specific computing design. Similarly trained analysts (e.g., three computing education researchers) might have missed some of these connections between the two fields, overlooking important aspects needed to gain a complete understanding of the design's role in computing education.

Despite these conflicts, the three analysts did agree that some computing learning objectives contained specific design skills, indicated by all analysts agreeing on a single skill code. To fulfill the learning objectives that correspond to the following themes, computing students likely must take on design roles.

Situating choices in the real world (UtC). All three analysts marked that learning objectives such as *"Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it"* (AP CS P 6.3.1) and *"Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts"* (AP CS P 7.4.1) corresponded to the design skill of *Understanding the Context*. Similarly, learning objectives like *"Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices"* (CSTA 2-CS-01) and *"Systematically design and develop programs for broad audiences by incorporating feedback from users"* (CSTA 3A-AP-19) indicate that students should be able to involve stakeholder perspectives in the process of creating their software. Learning objectives about contextualizing computing tended to occur in units about web development and game development/animation (Code.org), lessons about global impacts of computing and problem analysis (AP CS P), and standards about networks and the Internet or computing systems

(CSTA).

Generating ideas (CI). The analysts often found the design skill of *Creative Ideation* in objectives corresponding to generating many creative ideas or coming up with new concepts. For instance, deceptively simple objectives like *"Design the user interface of an app"* (Code.org 4.7) and more complex objectives such as *"Create a new computational artifact by combining or modifying existing artifacts"* (AP CS P 1.2.2) suggest that students should be able to generate creative, novel ideas for their software. Many objectives falling under this theme explicitly involved brainstorming as well, such as *"Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users"* (CSTA 1B-IC-19) and *"Brainstorm potential solutions to a specific problem"* (Code.org 4.6). Learning objectives involving generating ideas tended to occur in units about problem solving and user-centered design (Code.org), in lessons on creating computational artifacts (AP CS P), and in standards about algorithms and programming (CSTA).

Critiquing and evaluating tradeoffs (ES). Students learning computing with these objectives are expected to be able to evaluate how well software meets given or discovered constraints. The analysts unanimously identified *Evaluation & Synthesis* learning objectives like *"Test and refine computational artifacts to reduce bias and equity deficits"* (CSTA 3A-IC-25), *"Analyze the correctness, usability, functionality, and suitability of computational artifacts"* (AP CS P 1.2.5), and *"Critique a design through the perspective of a user profile"* (Code.org 4.2). Notably, computing learning objectives in the analyzed sets often expected students to evaluate the tradeoffs implicit in design choices, such as *"Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options"* (CSTA 2-IC-20), *"Evaluate computational artifacts to maximize their beneficial effects and minimize harmful effects on society"* (CSTA 3B-IC-25), and *"Consider the needs of diverse users when designing a product"* (Code.org 6.15). Learning objectives involving critique and evaluation tended to occur in units on data and society or user-centered design, (Code.org), lessons involving analysis of artifacts (AP CS P), and in standards about the impacts of computing or algorithms and programming (CSTA).

Incrementally updating software (II). Some learning objectives very obviously implied the design skill of *Iterative Improvement*. For example, analysts mapped objectives such as *"Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences"* (CSTA 1B-AP-13) and *"Iteratively improve upon a system for representing information by testing and responding to feedback"* (Code.org 5.2) unanimously to II. The analysts also saw iteration inherent in broader objectives like *"Apply a creative development process when creating computational artifacts"* (AP CS P 1.1.1). Learning objectives involving incremental updates to systems tended to occur in units about problem-solving or data and society (Code.org), in lessons involving abstraction (AP CS P), and in standards about algorithms and programming (CSTA).

Working with others (Comm). The need to communicate with peers surfaced in two main ways throughout each set of learning objectives. First, many objectives expected productive teamwork, evidenced in objectives like *"Communicate and collaborate with classmates in order to solve a problem"* (Code.org 1.1) and *"Collaborate in the creation of computational artifacts"* (AP CS P 1.2.4). Second, some objectives implied the need to become proficient at presenting and documenting design rationales, such as *"Communicate the design and intended use of program"* (Code.org 4.10) and *"Explain the design choices they made on their website to other people"* (Code.org 2.14). This second category of *Communication* is reminiscent of design specifications–documents drafted by designers that contain requirements for developers and engineers to implement (or for managers to approve). Occasionally the two categories overlap: *"Describe choices made during program development using code comments, presentations, and demonstrations"* (CSTA 1B-AP-17) includes both within-group communication (code comments) and external communication (presentations). Learning objectives involving working with others were common throughout each set of learning objectives. They tended to occur in units about human-centered design and web development (Code.org), lessons involving communication and collaboration (AP CS P), and standards about the impacts of computing, data analysis, and algorithms and programming (CSTA).

Types of Design Within Computing Education

While examining the results of the design skills analysis, I observed that there seemed to be more than one distinct type of design represented in the computing learning objectives. I performed preliminary affinity diagramming on the learning objectives that the original analysts identified as containing design skills to identify higher-level categorizations. Further collaborative discussion and refinement of the categories with all analysts revealed the following two types of design.

Problem-space design: The "what" and the "why". Problem-space design in computing answers the questions "What will this software do (or enable users to do) in the context of the world?" and "Why does the world need this software?" This form of design involves identifying, defining, and evaluating requirements for what an artifact is and what it should be able to do. The goal of problem-space design is to propose a solution to a particular problem which takes into account the constraints of the context as well as the views and requirements of stakeholders. In industry, requirements that reflect the problem space are generally created by project managers, product managers, or interaction or user experience (UX) designers. On the other hand, learners in computing education contexts typically lead their own projects or work with small teams. As a result, many students practice problem-space design to an extent. The learning objective *"Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users"* (CSTA 1B-IC-19) seems to involve only problem-space design, situating software requirements in the real world by expecting students to justify *why* certain software doesn't meet user needs and *what* that software might look like instead. Computing students might also practice problem-space design when they engage in work to define and update high-level software requirements, evaluate the fit of software against real-world constraints, or present their design rationale to peers or instructors.

Program-space design: The "how". Program-space design answers the question "How should the available tools and resources be used to effectively implement this software?" This form of design involves deciding how to meet requirements that result from problem-space design activities. Selecting the proper algorithms, data structures, and function calls to generate desired behavior are examples of program-space design. This kind of design is most often seen in software engineering and programming practices. In computing education, students writing code to meet requirements engage deeply in program-space design. Many of the objectives that seemed to involve only program-space design mirrored basic program design and software engineering skills (e.g., *"Develop an algorithm for implementation in a program"* (AP CS P 4.1.1)). Students could also practice program-space design without writing code, such as in "unplugged" activities where they create algorithms to solve problems. In this learning objective, students are not expected to justify *what* they are making or *why* they should address the given problem with an algorithmic approach: They are only expected to choose *how* to implement the algorithm so that it fits given requirements, though the choices they make while doing so still imply design decisions. Computing students might also practice program-space design when they decide which kinds of programming constructs to use in a program (e.g., conditionals vs. loops), or when they define and adhere to programming style guidelines to help communicate with teammates.

Interestingly, some learning objectives seemed to imply both problem- and program-space design practices. For instance, *"Iteratively improve upon a system for representing information by testing and responding to feedback"* (Code.org 5.2) seems to involve the problem-space skill of testing with users and applying feedback to refine higher-level *what* and *why* requirements. It also seems to imply program-space design in the iterative improvement of the program's code-level implementation to refine *how* the system works. The existence of this ambiguous space between problem- and program-space design might be one reason why the role of design has traditionally been so unclear and why (as suggested by my analysis in this section) it can be difficult to clearly identify. While program-space design might be considered an inherent, disciplinary form of design that exists within computing, problem-space design shares more characteristics with the discipline of design than it does with computer science or software engineering.

3.5 Study 2: How do problem- and program-space design manifest in K-12 computing education activities?

The results of Study 1 suggest that the nature of design and K-12 computing education's overlap is characterized by two distinct kinds of design: Problem-space design, in which students discover and define problem requirements; and program-space design, in which students decide the most fit way to meet the discovered constraints. These dual roles of design arose from analysis of learning objectives that imply certain computing proficiencies. However, what if learning objectives that contain design are simply ignored in practice? What if the curricula and standards we analyzed *do* contain design, but the design work is not instantiated in classroom activities? If the activities computing students participate in don't actually have design components, then the practical need for clarity around design and computing education's overlap becomes less pronounced. To address this possibility, I conducted a second deductive qualitative analysis on three sets of *student activities* from three K-12 computing education curricula.

3.5.1 Sampling Rationale: Confirming and Disconfirming Cases

To select the curricula for Study 2's analysis, I employed a confirming and disconfirming cases sampling strategy [235], as well as selecting another typical case for adequate coverage[5].

- Code.org's *CS Discoveries* [59] acts as my confirming case. In Study 1, I analyzed CS Discoveries' learning objectives for design skills; I should expect those design skills to be instantiated in the curriculum's student activities.

- The *AP CS A* course [62] acts as my disconfirming case, insofar as that is possible. In contrast to AP CS Principles' broad goal of increasing participation in computing, AP CS A is designed to mimic a university-level introductory programming course. Upper-level

[5]The Code.org CSD materials were the same as Study 1 for consistency and were collected on January 24, 2019 from the source found in the bibliography. The AP CS A and Exploring CS materials were collected on June 26, 2019. As before, my analysis reflects the published materials at that time, and does not cover updates between the time of analysis and the time of publication.

secondary students who take AP CS often intend to major in computer science in postsecondary education and can receive CS1 credit by passing the AP CS A exam. Due to the more technical focus of the course, we can expect its design-related activities to be more like discipline-specific design (program-space) than nondisciplinary (problem-space) design, potentially also occurring at a lower frequency than the other two cases.

- *Exploring Computer Science* [93] was selected as another typical case in order to ensure my analysis would encompass representative material. Exploring CS is a year-long, research-based curriculum intended to teach introductory computing concepts through inquiry-based activities in late secondary education. As of 2018, more than 50,000 students in 25 U.S. states and Puerto Rico had learned computing through the Exploring CS curriculum.

Exploring CS had explicitly labeled student activities within the lesson plans upon which I based my analysis. *CS Discoveries* also had labeled activities for the majority of their lesson plans; for the lessons which did not contain one or more explicitly labeled activities, I considered the entire lesson a single activity. The *AP CS A* course description provided sample instructional activities at the beginning of each unit. Though these were from the point of view of the teacher, they described students' activities, so I was able to use them in my analysis. Of note, the *AP CS A* curriculum guide did not provide sample activities for every lesson, which limits the findings of my analysis to a strict subset of all *AP CS A* student activities.

3.5.2 Analysis of Student Activities

Two analysts took part in the analysis of design in student activities:

- The dissertation author, the computing education researcher whose expertise is described in Study 1 (Section 3.4).

- A researcher and educator at a large, public, U.S.-based university with 10 years of experience in computing education research, 10 years of software engineering research and teaching, and 15 years of HCI + design research and teaching.

I used a deductive process to qualitatively analyze each student activity for the presence of design based on the two types of design in computing uncovered in Study 1 (Section 3.4.3) and the skills of design identified in the literature (Section 3.2.2). Each student activity could be marked with anywhere from zero (if no design were present) to ten (five problem-space design skills + the same five program-space design skills) codes. As before, I tended toward conservative interpretations of the student activities, relying on each activity's explicit accompanying text and descriptive materials to determine the presence and flavor of design. They also memoized each code with selection rationale.

To establish validity, the second analyst for Study 2 independently analyzed the student activity sets and marked codes upon which the two analysts disagreed, memoizing their rationale in a short comment. Then, the two analysts met to discuss discrepancies in their results and decide upon final categorizations. As mentioned in Section 3.4, I adhere to the perspective on qualitative coding presented by Hammer and Berland [132], treating the results of deductive coding as organizations of claims about data rather than quantitative data in themselves. As a result, collaborative discussion about disagreements led to refinement of the codeset. During their discussion, the two analysts noted the following major patterns in their disagreement:

- Applications of the *Understanding the Context* skill code for both problem- and program-space design differed based on whether one considered information gathering outside of the context of any particular problem to be a design activity (e.g., "Groups of students create lists of their ideas of what a computer is", ECS 1.1). Analysts decided that an activity should only be coded as UtC if there was a specific, defined problem the student was trying to solve, as opposed to discussing problems in general.

- Applications of various program-space codes differed based on whether one considered "unplugged" activities (which teach computing concepts without programming or computers) to be program-space or problem-space activities. After reviewing and discussing the definitions of problem- and program-space design, the analysts decided that program-space design work did not necessarily imply writing code and that unplugged activities could include program-space design.

- Applications of program-space *Evaluation & Synthesis* codes differed based on whether one considered critiquing tradeoffs of software created by an external person or entity to be a design activity in itself. Based on design literature, analysts decided that any evaluation of code for fitness to a certain constraint should be considered program-space ES.

- Applications of the *Evaluation & Synthesis* codes during presentation-based activities (e.g., ECS's gallery walks) differed based on how conservative the one was with their inference of activities. Analysts decided to limit their inference of activities and apply codes based only on the text of activity descriptions and supporting materials.

- Applications of program-space *Iterative Improvement* codes in debugging-focused activities differed based on whether one considered "improvement" to require implementing changes or simply proposing them. After discussing the core design skill definitions, analysts decided that debugging activities should have students actually implement changes to count as program-space II codes.

The two analysts collaboratively updated their final results to address these conceptual discrepancies. These clarifications also served to refine the definitions of the design skills and types of design in computing presented in this chapter. I then analyzed the results for patterns and themes.

3.5.3 Results

Characterization of cases

Table 3.1 summarizes the types of design skills that appeared in student activities by each unit of the three curricula.

As expected, the majority of Code.org CS Discoveries (CSD) activities included design: The analysts marked 100 of 150 activities as containing at least one skill of design. Of those 100, 52 only contained evidence of problem-space design skills, 37 contained only program-space design

	Problem-Space Design					**Program-Space Design**				
	UtC	*CI*	*ES*	*II*	*Comm*	*UtC*	*CI*	*ES*	*II*	*Comm*
Code.org CSD										
Problem Solving	✓	✓	✓	✓	✓	✓	✓			
Web Development	✓	✓				✓	✓	✓	✓	✓
Interactive Animation and Games	✓	✓	✓		✓	✓	✓	✓	✓	✓
The Design Process	✓	✓	✓	✓	✓	✓	✓		✓	✓
Data and Society	✓	✓	✓	✓	✓		✓		✓	
Physical Computing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AP CS A										
Primitive Types								✓	✓	
Using Objects							✓	✓		
Boolean Expressions and “if” Statements							✓	✓	✓	✓
Iteration							✓	✓		
Writing Classes								✓		
Array							✓		✓	✓
ArrayList						✓	✓	✓		
2D Arrays							✓	✓		
Inheritance							✓	✓		
Recursion								✓		✓
Exploring Computer Science										
Human Computer Interaction	✓	✓	✓		✓					
Problem Solving	✓						✓	✓		✓
Web Design	✓	✓	✓	✓	✓		✓	✓	✓	
Introduction to Programming	✓	✓	✓		✓	✓	✓	✓	✓	✓
Computing and Data Analysis	✓		✓	✓	✓					
Robotics	✓	✓	✓		✓		✓	✓	✓	✓

Table 3.1: The kinds of design skills that appeared in Study 2's analyzed computing education student activities. The leftmost column contains the names of units within each curricula. A checkmark (✓) indicates both analysts agreed that particular design skill appeared at least once in the unit's activities.

skills, and 11 contained both problem- and program-space design skills. In the Code.org CSD activities, problem- and program-space design practices were often tightly interleaved. As students created computational artifacts, they typically decided what they wanted to create and defined requirements for their artifact (problem-space design), then decided the most fit way to meet those constraints with programming or algorithmic constructs (program-space design), and finally implemented their plan. Frequently, the design work in both spaces was scaffolded by provided project guides which led students through activities like brainstorming, storyboarding, algorithm creation, and selection of programming constructs. Students often also justified their design rationale in both design spaces during peer review or formal presentations.

Design did surface in the AP CS A activities, much to the analysts' surprise. Notably, design practices in AP CS A were limited to the program-space: Out of the 36 activities, 29 contained program-space design, and none contained problem-space design. In software engineering terminology, students were often asked to come up with strategies to meet requirements, but they were never asked to define those requirements themselves. Generally, the program-space design activities centered around generating algorithms that fit certain given constraints (*Creative Ideation*) or analyzing programs for correctness or equivalence (*Evaluation & Synthesis*). Students occasionally worked in pairs and communicated their program design rationale as well.

Of the 288 Exploring CS activities, the analysts identified 96 that contained design: 46 problem-space only, 43 program-space only, and 7 containing both problem- and program-space design skills. Similar to the Code.org curriculum, Exploring CS activities often interleaved problem-space design and program-space design. Peer review, critique, and feedback (in both design spaces) were especially common in Exploring CS activities. Students often justified their design rationale and implementation choices to at least their teammates, if not the larger class. Often, they were asked to synthesize problem-space level feedback from users or peers into program-space design choices, then implement the results to incrementally improve their computational artifacts. Exploring CS activities also supported problem-space *Creative Ideation* very well by providing brainstorming guides where students wrote down multiple approaches to solving a problem with computational artifacts, selected one, and justified their selection before they implemented it.

Non-design activities

In activities where students were not actively solving a problem, they did not practice design. Often, these took the form of simple identification exercises such as *"For each [provided code] segment, have students trace through the execution of a loop with smaller bounds to see what boundary cases are considered, and then use that information to determine the number of times each loop executes with the original bounds"* (AP CS A 4.5). Non-design activities might also involve students simply manipulating data without necessarily having a higher-level goal in mind, such as in *"Groups create bar and mosaic plots with the data they have collected and additional contextual data sets"* (ECS 5.8). In the case of this last activity, the goal of the lesson was to have students become familiar with different types of plots, not to have them use those plots for any analysis. Similarly, some activities simply asked students to learn about general programming constructs and did not require application of that knowledge to solve a particular problem. For instance, the Code.org CSD activity *"Web Lab: Intro to CSS"* (Code.org 2.10) led students through a guided tutorial on how CSS can be used to change the appearance of HTML websites. While students would later use their knowledge of CSS to style their own web pages, this specific activity did not ask students to make design decisions–only to implement CSS styles as directed.

Exclusively problem-space design activities

The two analysts found instances of student activities containing only problem-space (nondisciplinary) design in both the Code.org CSD and Exploring CS curricula. When students perform these activities, they are working to define the "what" and the "why" of their artifact: motivating its existence and defining high-level requirements for its implementation. Problem-space design activities appeared in every unit of the Code.org CSD and ECS curricula, with most instances occurring in user-centered design (Code.org), human-computer interaction, and data analysis (ECS) units. Below, I present some themes that arose from final analysis results and indicate which of the five core design skills (Section 3.2.2) the group of activities tended to correspond to.

Research to motivate artifact creation (UtC and ES). One way in which problem-space design manifested in the activities was through information-gathering used to situate whatever computational artifact students planned to create in real-world contexts. Sometimes this took the form of user research, as in activities from Code.org's user-centered design unit: Activities *"Looking through a user's eyes"* and *"Responding to products"* activities (both Code.org 4.2) asked students to take the perspectives of different users to evaluate an existing product. After doing so, students identified opportunities for improvement in the existing products' designs. Another form this information-gathering could take was market research and data analysis. For instance, in the activity *"Brainstorming app ideas"* (Code.org 4.9), students analyzed applications already on the market to see how they addressed certain problems, then came up with an app idea that would differ from already-existing solutions. At other times, students sifted through data to characterize problems, as in the activity *"Groups do statistical analysis with mean, median, maximum, and minimum using the data they have collected and additional contextual data sets"* (ECS 5.10). ECS groups later used this data analysis to inform their final unit projects.

Requirement generation and refinement (UtC, ES, and II). Problem-space design activities sometimes had students come up with and iterate on requirements for artifacts they planned to create, rather than specifying all requirements in the student materials. For instance, the *"Define"* activity in Code.org's lesson 4.3 had students perform a version of stakeholder analysis to identify target user populations' needs. Similarly, *"Identify evaluation criteria and work in groups to evaluate websites using the rubric"* (ECS 1.3) asked students to identify requirements for determining the credibility of websites, then evaluate various sites against those requirements. Students later used these initial sets of constraints to inform project requirements.

Prototyping concepts and interfaces (CI and ES). Both curricula in which problem-space design appeared favored a design-before-implementation mentality, often instantiated in storyboarding and prototyping activities. Problem-space prototypes were generally low-fidelity, but all represented a concept for a final system. These systems did not necessarily have to be technological: For instance, in the *"Create a representation"* (Code.org 5.8) activity, students designed a paper punch card with categories of information they would use to represent their "perfect days." However, when the system was technological, storyboarding and prototyping activities often helped students envision user interfaces, such as in the *"Paper prototypes"* activity (Code.org 4.7), where students created and tested user flows of their apps by sketching screens on notecards. Sometimes, students were led to consider user perspectives other than their own when prototyping: The notes for the *"Create a storyboard for a multipage web site"* (ECS 3.8) activity suggest that teachers prompt their students to consider the many kinds of diverse users that might use their website, though students are not evaluated upon this requirement.

Managing group expectations and decisions (Comm, CI, and ES). When doing group work, teams often practiced communication around problem-space design decisions and how their group would meet particular design requirements. Both the Code.org CSD and Exploring CS curricula contained some form of team management activity such as *"Groups discuss roles and responsibilities"* (ECS 5.3) and *"Team contract"* (Code.org 4.8). These kind of activities focused on how the team would effectively work together to create a final computational artifact. Sometimes groups worked together to generate a single set of design requirements, such as in *"Building an Aluminum Boat"* (Code.org 1.1), where teams came up with different ways to create boats from aluminum foil that would float in water when weighed down with coins. Other times, teams had to come to a consensus on which requirements to prioritize and implement. For instance, teams came into the activity *"Groups work to merge individual data sets together"* (ECS 5.1) with data they had individually collected (with no particular constraints on how to categorize it) and were asked to merge the data sets together. In doing so, groups had to agree upon which dimensions of their disparate data sets to change so that the data in the final set ended up at the same granularity, which inherently involved making decisions about what data was important. Other times, team consensus-building happened in response to feedback: Using the results of *"User Testing"* (Code.org 4.11), teams discussed and prioritized feature implementations and bug fixes. Problem-space communication underlies all these activities and helps students to successfully create artifacts while working with teams.

Justifying problem-space design choices (Comm). Finally, when practicing problem-space design, students were often asked to justify why their solution fit a particular problem. Note that this is different than discussing *how* they implemented their solution (which would represent problem-space design justification). For example, in the *"Student teams present projects"* (ECS 1.2) activity, pairs of students worked together to explain their rationale for giving a particular user a specific computer hardware configuration (based on the fictitious person's typical usage patterns). During the *"Presentation prep"* and *"Presentations"* (both Code.org 4.16) activities, students wrote up and presented thorough design specification documents, including their problem motivation and framing, existing solutions and their shortcomings, examples of how their artifact addressed these shortcomings, and their user testing refinement process.

Exclusively program-space design activities

Instances of activities containing only program-space (disciplinary) design appeared in all three curricula. When students perform these activities, they determine the "how" of their artifact: given some problem-space constraints, how at the algorithmic level those constraints should be implemented. Program-space design activities occurred in every unit of the Code.org CSD and AP CS A curricula, though it was absent from two units of the Exploring CS curriculum (human-computer interaction, and computing and data analysis).

Understanding programming environment affordances (UtC). Students often worked to understand the kinds of functionality they had available to them before they began to implement their artifacts. This is different than simply learning about programming language constructs (e.g. *for* loops or arrays), which are general concepts that underlie all programming practice. Instead, students exercising this kind of program-space design skill learn about unique features that are not necessarily transferable to other programming environments. For instance, in activities like *"Intro to Sprites"* (Code.org 3.6) and *"The Draw Loop"* (Code.org 3.7), students explored the functionality of Code.org's Game Lab platform. Later, they used their knowledge of Game Lab's affordances to create interactive animations and games. Similarly, in the *"AppLab exploration"* (Code.org 4.10) activity, students explored implemented apps in the AppLab environment in order to understand the functionality available to them. In the *"Pairs investigate features of Scratch and start name assignment"* (ECS 4.1) activity, students experimented with Scratch blocks to figure out what they could accomplish within the environment. Each of these activities is a form of the program-space skill of *Understanding the Context*, teaching students the affordances of their specific environments or IDEs.

Designing algorithms and program behavior (CI and II). A common form of program-space design was devising algorithms or structuring programs to fit given constraints. Sometimes, the constraints were defined by the teaching materials and simply required iterative modification of a base program, requiring students to figure out the best way to meet those constraints. This occurred in activities like *"Provide students with a method [with some functionality]... ask students...to write a similar method that, given a student number as input, returns the name of a student from a String containing the first name of all students in the class, each separated by a space"* (AP CS A 4.1-4.4). At other times, requirements were defined for students, but they had to figure out how to meet them without a base example, as in *"Develop an Age program"* (ECS 4.9), where students created a program that responded to different numerical (age) inputs in various specified ways, and *"Making music"* (Code.org 6.11), where students used a physical circuit board to create audible buzz patterns. Notably, these kinds of activities did not necessarily include writing code: *"In groups, participate in the candy bar activity"* (ECS 2.2) had students come up with an algorithm to split a candy bar in the least amount of cuts. Students also practiced problem-space design when they created code architectures that fit a particular set of design requirements, as in the activity *"Given a class design problem that requires the use of multiple classes in an inheritance hierarchy, students identify the common attributes and behaviors among these classes and write these into a superclass ..."* (AP CS A 9.2-9.4).

Program analysis and refinement (ES and II). Students often analyzed programs for correctness or equivalence. This kind of program-space evaluation activity was particularly prevalent in the AP CS A curriculum, comprising the majority of the curriculum's design activities. In the AP CS A curriculum, students often had to compare actual output against expected output, then propose and implement fixes to align program behavior with requirements. For example, in *"Provide students with code that contains syntax errors. Ask students to identify and correct the errors... [and] have them verify their conclusion by using a compiler and an IDE that does not autocorrect errors"* (AP CS A 1.1), students identified and fixed syntax-level errors. Students also evaluated code against requirements at the semantic level, as in *"Provide students with several error-ridden code segments containing array traversals along with the expected output of each segment. Ask them to identify any errors that they see on paper and to suggest fixes to provide the expected output..."* (AP CS A 9.4). Debugging and refinement activities appeared throughout all three curricula, in activities like *"Web Lab: Smash those Bugs"* (Code.org 2.8), *"Enhance the variable example"* (ECS 4.9), and *"Test the robot frequently and refine program and hardware."* (ECS 6.10). Program analysis activities often also asked students to determine equivalence of statements (*"Provide students with a code segment that utilizes conditional statements and a compound Boolean expression, and ask them to choose an equivalent code segment that uses a nested conditional statement ..."* (AP CS A 3.6)). Finally, students participated in program-space design evaluation when comparing different algorithms: in *"Model the tower building algorithm"* (ECS 2.6) and *"Groups participate in the various parts of the CS Unplugged: Lightest and Heaviest activity"* (ECS 2.7), students evaluated the tradeoffs of using particular algorithms over others, then selected the best fit for their situation.

Justifying implementation and code-level teamwork (Comm and ES). When working in groups, students had to communicate with others about program design decisions in order to build functional programs. The Exploring CS curriculum almost always had students working in pairs or groups when creating software. In the curriculum's Intro to Programming unit, activities like *"Pairs complete Map Route Activity"* (ECS 4.4), *"Develop Social Media Quiz program"* (ECS 4.10) , and *"Create a timer block with a parameter"* (ECS 4.14) all required students to work with at least one partner. In doing so, students often discussed and justified their program-space design decisions with peers. These kinds of discussions were explicitly scaffolded by project guides in the CSD curriculum (Code.org 3.22 and 6.16), where students collaboratively decided upon program behavior and generated pseudocode before they began implementing their solutions. Pair programming activities were prevalent in all three curricula (e.g. *"Have students use pair programming to solve an array-based free-response question. ..."* (AP CS A 6.4); *"Pair programming"* (Code.org 2.4)). Sometimes pair programming activities involved evaluation and critique, as in *"Provide students with the pseudocode to multiple recursive algorithms, and have students write the base* *case of the recursive methods and share it with their partner. The partner should then provide feedback, including any corrections or additions that may be needed"* (AP CS A 10.1). Notably, Code.org's CSD activities often embedded instruction about code style best practices into students' introductions to programming concepts. For instance, the activity *"Programming with variables"* (Code.org 3.5), students' first exposure to variables, instructed students on and had students adhere to style guidelines for variable naming and commenting. Subsequent activities encouraged students to keep up these practices in order to facilitate better within-team communication about program implementation choices. Finally, similar to the previously mentioned problem-space practice, students often had to justify their implementation choices, describing how their solution met (or enabled them to meet) requirements. During whole-class discussions or demo days (e.g. *"Participate in discussion of solutions"* (ECS 2.2), *"Groups present final projects"* (ECS 2.9)), students often described and defended their program-level design choices to peers.

Overlapping problem- and program-space design activities

The two analysts identified a handful of student activities in which both problem- and program-space design overlapped in the Exploring CS and Code.org CSD curricula. Students tended to practice both kinds of design in tight iterative cycles in activities fitting this categorization, drilling down through requirement definition to algorithm design and finally implementation, then popping back up into a design space to either further refine or gain new perspectives on their computational artifacts. Activities which contained both problem- and program-space design appeared in five of the six Code.org CSD units, with the majority in *Interactive Animations and Games* and *Physical Computing*, and three of the six Exploring CS units, with the majority in *Intro to Programming*.

Understanding existing solutions at multiple levels (UtC and CI) Sometimes, students were asked to understand both the problem and program-level design choices inherent in existing solutions to a particular problem, both defining and ideating on requirements. For instance, at the beginning of the Physical Computing unit, Code.org students participated in *"Innovation research"* (Code.org 6.1). During this lesson, students researched existing physical computing devices and answered questions like "What problem does it solve?" (a problem-space design question), "How do you interact with it?" and "How could you improve it?" (both program-space design questions). Later, they used these findings to inform their own robot creations. Similarly, ECS students practiced both kinds of design in *"Participate in Rock Paper Scissors discussion"* (ECS 4.11). In this activity, students first worked to define the program-space requirements for a game of Rock Paper Scissors, then (with help from a partner) defined an program-space pseudocode algorithm to determine the winner of a given round of the game. Throughout both these activities, students practiced each kind of design.

Simultaneous evaluation in both design spaces (ES and Comm). Occasionally, when students evaluated each others' work, they critiqued both how well the artifact fit its requirements (program-space) and how well those requirements represented the problem at hand (problem-space). One example of this is found in a lesson where students created interactive greeting cards with animations: During the *"Peer review"* portion of the assignment (Code.org 3.14), students critiqued others' cards. In the program space, students evaluated the implementation of the card – ensuring that multiple properties were updated in the animation rendering loop, that the code was properly modularized, and so on. During the same critique, they also evaluated how well the card met problem-space requirements, such as ensuring that it responded to multiple kinds of user input and suggesting ideas for future problem-space requirements. Similar dual critiques occurred in the ECS activity *"A member of each group will demonstrate and explain a program modification to their assigned gallery walk group"* (ECS 6.6).

Improving artifacts by implementing feedback (II and ES). When implementing feedback from critiques and user testing, students practiced problem-space design when they synthesized the feedback into new requirements, and program-space design when they extended the functionality of their artifact to meet the new requirements. The activity *"Fixing bugs and adding features"* (Code.org 4.15) asked students to incrementally improve their prototypes after a round of user testing. Bug fixes almost certainly involved program-space design to patch the code, while new feature implementation likely included both defining requirements and writing code to meet them. Code.org students responded to user feedback in both design spaces in multiple activities throughout the curriculum (e.g., *"Prepare [to finalize an artifact]"* (Code.org 5.15), *"Iterate -*

revise prototypes" (Code.org 6.16)).

Project planning and implementation (all skills). Finally, the overlap of problem- and program-space design showed up most often in final project activities. Often, these projects occurred at the end of a unit and were intended to showcase students' learning as they worked in teams to define and motivate a problem, plan the design of a computational artifact to address the problem, then implement, test, and refine their solutions. For example, *"Develop a Create Your Own Adventure animation"* (ECS 4.5) asked students to practice a wide swath of design skills. In this activity, students worked in pairs to create a multilevel interactive adventure game. In the problem space, pairs storyboarded their adventure before implementing it, creatively generating requirements and discussing design rationale for their levels. In the program space, students attempted to meet these requirements by implementing event handlers for user input (exactly how they did this was left up to students to decide), evaluating and updating their code until it met the problem-space constraints, and communicating while pair programming. Most project planning activities also contained both forms of design, even if the actual implementation of the design was left to a later activity. For instance, the *"Unplugged: program planning"* activity (Code.org 6.9) asked students to create some sort of interactive game with physical computing hardware. The problem-space requirement definition process was left entirely up to student groups, along with any program-space ideations around how to respond to user input. When students implemented their plans, they iterated on their solutions until they felt like they were ready to test it with peers, at which point they facilitated user testing and improved their prototypes based on the results. Another project activity *"Use the planning document to plan the robot"* (ECS 6.11) had students go through a similar process to create an autonomous "search and rescue" explorer robot. Each of these project activities had students acting as both designers and developers as they moved through the full artifact creation process.

3.6 Discussion & Concluding Remarks

My analysis in the previous sections suggests that there are design skills in many of our most widely disseminated computing education standards and the activities that stem from them. Table

	Problem-Space Design	**Program-Space Design**
	Why does the world need this software? *What* should this software do (or enable users to do) in the world?	*How* should this program be structured? *How* should I implement this software's requirements most effectively?
Understanding the Context	• Requirement generation • User research • Market research • Exploratory data analysis	• Learning environment affordances • Understanding existing code • Decomposing problem into modules
Creative Ideation	• Storyboarding • Brainstorming possible solutions • Sketching and prototyping	• Creating algorithms • Writing pseudocode • Defining code architecture
Evaluation & Synthesis	• Using feedback to inform requirements • Critique • Evaluating solution against constraints	• Verifying correctness and equivalence • Analyzing implementation tradeoffs • Program behavior analysis • Code reviewing
Iterative Improvement	• Refining problem conception • Updating requirements	• Debugging • Refactoring code-level architecture • Extending existing code
Communication	• Presenting design rationale • Collaboratively defining requirements • Managing group roles/ responsibilities	• Presenting implementation rationale • Adhering to style guidelines • Pair programming

Table 3.2: Examples of problem-space (nondisciplinary) and program-space (disciplinary) design activities in computing education contexts, organized by the core design skill they correspond to. Each of these activities appeared in at least one of the analyzed computing curricula. While I present these activities in two clean columns, I note that some activities may exist in the intersection of problem- and program-space design depending how they are implemented in a particular classroom or course (e.g. "Creating algorithms" might require students to draw both on implementation knowledge and knowledge about the world).

3.2 summarizes some the kinds of design-related activities in computing curricula, broken down by their alignment with the five core skills of design from Section 3.2.2 that served as my basis for analysis. The skills identified include both *disciplinary* forms of computing-specific design such as devising and refining algorithms (referred to as program-space design), but also skills from the broader discipline of design, which involve understanding and situating software in the world (referred to as problem-space design). While problem-space design activities tend to align more with most people's conception of general (i.e. not discipline-specific) design, program-space design activities seem closely linked to computing in a way that suggests they are more aligned with discipline-specific computing design. As evidenced by my Study 2 results, there were also a number of computing education activities that contain both kinds of design, either simultaneously or interleaved with each other.

As it stands, many of our K-12 computing education materials mask the presence of design in computing education by either simply ignoring it or by claiming many design practices and skills as computing proficiencies. However, continuing to do this might be detrimental to teaching and learning in a number of ways. Students learning computing through design-laden curricula may mistakenly interpret design practices as computing-specific skills. Though this is appropriate and authentic for program-space design practices, these students would likely fail to recognize the existence of problem-space design and distinguish it from computing. They might then be misled into pursuing careers in software engineering when their interests are more suited to user experience design or product management, for instance. Educators may also suffer from a lack of clear boundaries between design and computing, which complicates teaching. Prior work has shown that the pedagogical content knowledge (PCK) required to teach design is meaningfully different than the PCK required to teach computing [146, 228, 229], since PCK is domain-specific [137, 266]. The PCK needed to teach problem- and program-space design might also differ in meaningful ways. Any effort teachers spend figuring out how to teach computing skills may not transfer if the skills they teach are, in fact, design.

If the status quo is problematic in these ways, how might this overlap be addressed? One possibility would be to try and remove nondisciplinary problem-space design (the "what" and the “why”) from computing education, while retaining disciplinary program-space design (the “how”). The result of this would be a set of competencies that are solely concerned with computing concepts and the design skills required to architect, plan, construct, and refine programs. Such work, however, would be difficult, time-consuming, and likely even counterproductive to the goal of training well-rounded computing students. As Section 3.5 demonstrated, many nominally computing-related learning objectives *only* concerned problem-space design, while others contained both problem- and program-space skills. Pragmatically, we probably should not spend time and resources to undo the work that has already gone into creating and defining computing curricula.

Even if we decided to move ahead with the work of fully separating design and computing in educational contexts, prior work suggests that aspects of problem-space design are important for understanding how computing connects to the world. For instance, those who create software *should* understand that there are differing software usage styles, and that the software they create will almost certainly be used by stakeholders who are not like them [107,283]. Computing literacy *should* also extend to understanding the ethics and justice issues inherent to computing and the impact that computational approaches can have in society [55,254,286]. In the same vein as Chilana et al.’s *conversational programmers* [53], who learn to code not to become developers, but so that they can communicate better with developers, computing education might aim to educate computing students who are *conversational designers*, able to communicate effectively with designers in their teams and organizations.

Problem-space design skills are also often used in computing education to motivate students to engage with computing material (see Section 3.1). Removing the *what* and *why* from computing education, leaving only the *how* might harm interest and engagement in computing education. While demand for all occupations is projected to grow 7% by 2026, the demand for computing-related occupations is projected to grow by 19% [87]. If we are to have a chance at meeting the demand for a computing-literate workforce, we cannot afford to dissuade learners early on by focusing solely on the design of programs and not on their use in the broader world. Eliminating problem-space design might even lower the overall diversity of the computing field: Prior work suggests that situating computing within creative or design-based topics can encourage women [97, 161, 236] and underrepresented ethnic groups [77] to engage with computing where they otherwise might not.

If we can't ignore design's presence in computing, and since it seems counterproductive to try and excise non-disciplinary problem-space design from computing materials, another option is to embrace the entanglement of design and computing and attempt to better understand the two fields' unique intersection through further research. Given the evidence presented in this chapter and the arguments presented in prior work [302], there is a meaningful interaction between design and computing education which would be well-served by further scrutiny. As mentioned in Section 3.3, other fields that overlap with design have found success in this approach. For example, by recognizing design as a distinct yet necessary part of K-12 engineering education, Crismond et al. created a framework of "misconceptions, learning trajectories, instructional goals and teaching strategies that instructors need to know to teach engineering design effectively" [73]. In architecture education, one of the earliest disciplines to recognize and study its overlap with design education, Lawson studied how students acquire design skills in order to suggest improvements to computer-aided design (CAD) systems [186]. A branch of research specifically studying the intersection of computing education with design could produce similarly fruitful and interesting results.

Toward this end, future work should build upon the distinctions, intersections, and opportunities identified in this chapter to further investigate how to effectively integrate design and computing education. Such work might reveal how design and computing interact not only in the program space, but also in the largely unexplored problem-space design of computing artifacts. For example, studies might begin to reveal how difficulties that students face with debugging (part of program-space design) might lead students to change their requirements (part of problem-space design), or how constraints discovered while designing an algorithm (program-space design) reveal insights about a program's value in the world (problem-space). Investigating design and computing integration would also enable us to draw upon both fields' theoretical and pedagogical bases. For instance, *Iterative Improvement*, a design skill described in Section 3.2.2, sits squarely in this intersection.

Researchers might ask questions like *What are the most effective pedagogical strategies to teach iteration and the notion of productive failure in computing contexts?* Design education pedagogy could inform particular approaches to the question, while computing education work might suggest adaptations and pitfalls specific to computing contexts, leading to unique results that inform about the overlap of design and computing.

Since the intersection of computing education and design is not yet well explored, understanding starts with the CER community and future exploratory work. Embracing a research area that recognizes the nuanced role of design in computing opens up many new exciting avenues of investigation. For example, what are concrete strategies to address design-related learning barriers students in computing education concepts (such as those recently described in my previous work [229])? For students in computing classes who decide they are more interested in design than computing, how should we effectively route them to design disciplines? How can learning be structured to isolate program-space design from problem-space design? What are effective ways to teach design-related skills like perspective-taking, communication, and interpreting user feedback? And what kinds of pedagogical content knowledge do educators need to effectively teach design in computing contexts? There also fascinating questions about learner identity. How can explicitly naming design's presence in computing and teaching design skills change students' perceptions of the field and their roles within it? Can embracing design in computing draw more students to the field who do not fit into traditional computer science stereotypes, as has been shown in some prior work [77, 97, 236]? By answering these questions, we may be able to more effectively leverage design to effectively and equitably engage diverse learners.

One of the first places the above kinds of research might appear is in curriculum and instructional design. Recognizing and identifying the presence of problem- and program-space design in computing might require the reframing of existing materials as well as creation of new materials. For example, existing materials that already contain design skills (such as those studied in Sections 3.4 and 3.5) may require a framing update. Resources might clearly distinguish and name both design and computing skills within lesson plans, so that the teachers using them know how to prepare. A potential example of this would be something similar to the tags used in Code.org's curricula (which inform the teacher at a glance if the lesson's material requires a physical computer, uses the Game Lab platform, etc.). In K-12 education, materials might even provide explicit instruction on the role of design within computing and the two disciplines' overlapping natures. For instance, content creators could include more information about fields in the intersection of design and computing and explicitly name them (e.g., "If you liked this lesson, consider careers in interaction design."). Some of this work has already begun. Since the time of my analysis in Sections 3.4 and 3.5 (early to mid-2019) and the time of publication, Code.org's CS Discoveries curriculum has already modified their unit on the design process to explicitly integrate more information about user-centered design best practices and empathy with diverse kinds of users, among other updates.

These shifts in curriculum come with their own changes to teacher preparation. Pre-service and professional development on K-12 computing education might need to explicitly distinguish between computing and design, sharing the concepts in this chapter and revealing how they play out in classrooms. Teachers may need to learn to discriminate between computing skills and design skills (some initial work on this topic is described in Section 3.3.4). They may also need to recognize and respect the fact that students can have differing motivations for learning computing, some of which may lead to design professions rather than computing ones. Teachers may also need to understand that a student being proficient at computing concepts does not mean that they will also be proficient at design concepts (and vice versa). Higher education faculty may also need to embrace these concepts, suggesting a need for structural change. For example, embracing the role of design might involve making the field of human-computer interaction more central in computer science curricula (as opposed to its current role as a popular elective). It might also involve changes in how the rest of computing is taught, leveraging design's studio approaches [52, 80, 247], ideas of productive failure and iteration (e.g. [16, 239]), and approaches to teaching [78, 154, 183] and assessing [80, 85] creativity and prototyping processes. In these ways, teachers at all levels can use design to broaden the definition of computing while simultaneously respecting its role as a distinct discipline.

In addition to teachers, there are many opportunities for school administrators to embrace design more systematically. Some secondary schools in the United States, for example, already teach graphic design courses, but these courses are often siloed to arts departments and disconnected from computing education. K-12 administrators could support leveraging both design and computing in these respective learning contexts. Such administrative changes might also require higher-level policy change that provides more comprehensive definitions of computing that recognizes the distinct role of design.

As with any work that suggests directions for the future of computing education, there are some intrinsic limitations to what can be realized in practice. First, primary and secondary curricula coverage for any area are necessarily constrained by limitations of time, and computing education is no exception to this. Deciding which computing topics to prioritize over others in order to better integrate an understanding of computing design becomes a question of curriculum design. My results uncovered and characterized the presence of design *within* established computing curricula, suggesting that implementing the changes discussed above might be less of a full excision of particular topics and more of a change to the framings and approaches we use to better represent the design that is already there. The goal of this chapter is not to provide a definitive answer as to which computing topics should be cut in future curriculum, but rather to discuss and provide insights into what computing education might gain if it were to prioritize making the presence of design more explicit. In this way, my contribution is similar to recent work that suggests improving computing education by approaching it from perspectives like sociopolitical identity [288] or culture [77]. Another limitation is that K-12 educators and administrators will have questions about the what the results presented in this chapter imply for their particular contexts and courses, some of which we can not yet answer definitively. As discussed in Section 3.3.4, teaching and learning design within computing contexts is difficult for a number of reasons, and while recent work has begun to explore this space, much of it focuses on postsecondary education. We still need to develop pre-service preparation programs, useful materials, and concrete pedagogical strategies for teaching computing design topics. These areas provide directions for future research on design and computing's overlap in primary and secondary educational contexts.

While there are challenging choices to make about computing curriculum, acting on these implications in some way is critical to help students gain a holistic understanding of computing in the broader world. Understanding the ways that design manifests in computing contexts—in decisions about how software fits into the world and how to implement real-world requirements—enables computing students to better recognize and respond to pervasive embedded biases in software [18, 70, 178], whether they are the software's creator or simply a user who experiences its effects. Because even basic changes in deploying computing education across the globe are already challenging, many of these implications may seem intractably hard. On the other hand, changing computing education infrastructure, such as curricula, programs, and policy, will never be easier than it is now, where there is little inertia, much momentum, and immense opportunities for literacy.

Chapter 4

COMPUTING STUDENTS' LEARNING CHALLENGES IN HCI EDUCATION

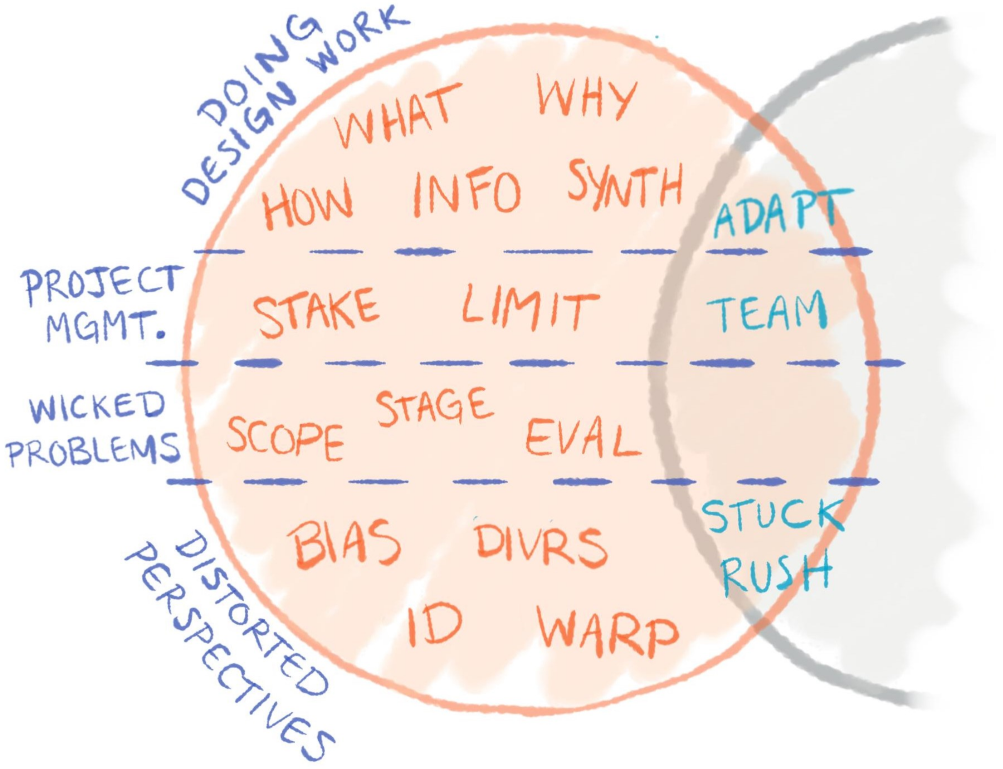


Figure 4.1: The study described this chapter found four categories of problem-space design learning challenges computing students might encounter, some of which also involved elements of program-space design.

4.1 Introduction

In higher education computing programs, students are often taught to *engineer* software. However, developers often make *design* decisions that impact the usability, accessibility, and inclusiveness of their software, including at companies that lack design cultures [174], startups that lack designers [174], in open source projects without design workflows [176], and even at large companies where they manage or collaborate with design teams [193]. In each of these settings, understanding user experience (UX) and interaction design concepts is key to creating and collaborating on high-quality software.

Unfortunately, partly because many developers lack design literacy, software still routinely fails to be usable for diverse populations (e.g. [26, 44, 126, 177, 228]). Professional software engineers still struggle with design-related tasks like requirements elicitation [1] and interface creation [237]. Since design choices are not value-neutral, poorly designed software can unintentionally perpetuate harmful stereotypes [32, 33, 254] or disadvantage already-marginalized populations [44].

At the heart of this problem is the fact that many developers receive little to no design training before entering the workforce: Students in traditional computer science (CS) degree programs may take at most one human-computer interaction (HCI) or interface design class prior to graduation. Even when computing students take an HCI class, teaching them design skills is hard [302]. Educators often struggle to engage students [144, 206, 247], to override persistent perceptions that designerly aspects of HCI are "inessential" [55], "easy," or "commonsense" [91], and to accurately assess students' design work [31, 282, 316]. Additionally, much of this research is limited to educators' reflections on their own particular courses [192], so students may face challenges that educators do not perceive.

The goal of this chapter is to understand what computing students struggle with when learning to design software interfaces in order to inform HCI pedagogy. My working definition of HCI design in this chapter draws on the model described in Park and McKilligan's review of HCI design and design thinking [233], which focuses heavily on software interface design as a basic HCI design proficiency. For this exploratory study, I scope my investigation to software interface design learning, and I use the broad term *learning challenge* to represent anything that prevents a student from effectively learning or applying software interface design concepts in school or as a practicing software developer. My intended audience consists of higher education instructors who teach software interface design principles within computing-focused (rather than design-focused) departments, as well as researchers who study formal and informal HCI design learning contexts.

In this study, I ask the research question *What challenges do computing students face when learning and applying software interface design skills?* To answer this, I qualitatively analyzed survey responses (n=117) and interview transcripts (n=15) from computing students learning design skills in both formal (post-secondary classes) and informal (on-the-job) settings. From this, I identified 15 types of learning challenges. I then validated the set through surveys (n=35) and follow-up interviews (n=8) with HCI educators who teach software interface design concepts to computing students. From educators' responses, I identified a further 3 types of difficulty, resulting in a final set of 18 learning challenges. This set of student challenges provides a potential basis for improving HCI design education. Finally, I discuss implications of this study for future research on HCI design pedagogy and educational practice. My contributions are: [1] [2] [3]

1. A set of challenges computing students may face when learning and applying software interface design concepts, with examples; and

2. Validation of these challenges from educators who teach interface design to computing students, with examples.

[1]The original publication used the term "student learning difficulty" throughout to describe the barriers to student learning I investigated. However, since then, I have learned that the term "learning difficulty" is sometimes used as coded language to minimize learning disabilities in educational contexts. Therefore, in this chapter, I have updated the terminology to "student learning challenge" to better reflect the original intent of the study. Thanks to Jayne Everson for pointing this out.

[2]Throughout this chapter, I use first-person language ("I", "my", etc.) to describe this work to reflect the single-author nature of the dissertation document. However, this work was conducted in collaboration with Meron Solomon and Dr. Amy J. Ko, and published in the 2020 ACM CHI conference on Human Factors in Computing Systems [229]. I acknowledge the shared contributions of all authors to the original publication.

[3]This work was supported by NSF grants DGE-1762114, 1735123, 1539179, 1703304, and 1836813.

4.2 Related Work: HCI & UX Pedagogy

HCI has many of its roots in traditional CS programs, though it encompasses many disciplines. Faiola's Design Enterprise Model (DEM) [94] situates design, especially interface and interaction design, as a core competency for students learning HCI principles. Design skills are known to be difficult to teach and learn within HCI education contexts [94, 268, 302]. As discussed in the introduction, computing students often fail to engage meaningfully with the designerly aspects of HCI [144, 206, 247] or erroneously view design as easy, inessential work [55, 91]. HCI educators, who (especially in computing-focused departments) may not have design backgrounds, often struggle to assess design-related work adequately [31,282]. HCI educators often face a time crunch in already overcrowded computing curricula as well, [55, 130], forcing educators to prioritize some topics and exclude others, which leads to wide variation in HCI coursework and course content.

To address these challenges, prior work has explored the efficacy of the studio approach for HCI design topics. Studios are intended to be "bridges" [28] between education and professional settings, providing a semi-authentic environment for students to develop skills they will use in future workplaces. While this body of work has provided valuable insights into how an HCI classroom can be structured, computing students may experience difficulty learning in studio environments due to their unstructured nature [147, 247]. Educators have also reported significant challenges successfully teaching studio-based classes [206], at least some of which were centered around motivating students to engage with concepts.

The related field of user experience (UX) pedagogy has explored the space of teaching and learning software interface design more extensively, often with potential implications for HCI education. Getto and Beecher drew on their combined expertise from teaching and industry to propose a model for incorporating UX education into existing higher education curricula [109]. Gray and colleagues' work on UX competency also provides recommendations for UX pedagogy which may transfer to HCI contexts. For instance, Gray's analysis of how student designers transition into professional communities indicates that perceptions of competence are often situated in a particular context and individual experience, which implies a need to focus on designerly identity formation and working within organizational constraints in UX higher education [124]. Gray et al. subsequently proposed a model of the interactions that occur between individual and group competence in professional UX contexts: Individual knowledge (often gained through formal education) forms a basis for individual competence, but it also interacts with informally gained knowledge which comes from the group or organizational context [125]. Both formal and informal learning play a role in UX competence under this model, which suggests a need to study learning challenges in both contexts to gain a complete picture of students' educational experiences.

In both HCI and UX pedagogy, prior work has leveraged the frame of learning challenges to gain insights into how to improve education. This kind of framing is beneficial to both students and educators [268]. Getto and Beecher's investigation identified the presence of significant institutional barriers to implementation of UX education and provided strategies for surmounting these barriers by calling for increased partnerships between academia and industry, though it did not directly address *student* learning challenges [109]. Siegel and Stolterman framed their study of non-designers transitioning into design roles around a set of observed student learning barriers, which they drew upon to propose a framework for designing instructional activities for HCI design classrooms [268]. However, this particular study drew on the experiences of students of many different educational backgrounds taking part in a design-focused program. In contrast, my investigation seeks to understand the experiences of students with a specific (computing-based) educational background enrolled in a program where design is not the main focus. I seek to extend this body of work which uses student learning challenges as a focus to inform HCI education around software interface design principles.

4.3 Study 1: Student Perspectives

4.3.1 Method: Surveys and Interviews

To begin identifying the kinds of challenges computing students face when learning software interface design skills, I collected and analyzed data from both surveys and interviews.

Surveys of Students Formally Learning Design

I surveyed undergraduate computing students enrolled in introductory software interface design classes at two large, public, U.S. universities:

- Computer science students enrolled in a course at University 1 (U1A), which focused on usability principles. This course was only mandatory for students enrolled in a non-default option within their major, and was generally taken in students' 3rd or 4th year.

- Information science students enrolled in a course at University 2 (two sections – U2B and U2C) focused on design methods with a human-centered design model. This course was mandatory for all students in the major and typically taken in students' 2nd year or later. I considered the information science students at University 2 to have computing backgrounds due to the highly technical nature of their program and its strong emphasis on software development.

Due to the sequencing of the universities' programs, this was likely the first formal exposure to software interface design concepts for many students.

The surveys I used at each university contained similar items, adjusted only to fit the particular class's context. I prompted students to "*Write down any questions you still have about prototyping [or other relevant design topic] after today's lesson.*" I elicited learning challenges by asking for students' questions about design-related topics (as opposed to asking them to directly report the nature of their struggle) because students may not have had the language or awareness to accurately identify the causes of their confusion, especially if they were new to design work and unfamiliar with terminology.

I also collected demographic data to verify students' fit with the population of study, including students' self-reported fields of study (to verify they were computing-related) and their self-reported experience with designing interfaces (to verify their novice status). I did not collect further demographic information (e.g. gender, race/ethnicity, age) both because I wished to keep the survey lightweight and because I did not believe it to be relevant to the topic of study during this initial investigation. At University 1, I administered the surveys electronically via a link sent to a learning management system; At University 2, a researcher attended the classes and administered surveys to students on paper.

From these three courses at two universities, I collected 117 responses (U1A: 13, U2B: 33, U2C: 71) representing perspectives from 88 students (U1A: 9, U2B: 41, U2C: 38) since some students wrote multiple questions.

Interviews with Students Informally Learning Design

The perspectives collected through the above surveys represent learning challenges experienced in formal educational contexts. However, learning can also occur in informal contexts, such as on the job or while working on software design projects for external clients. I therefore conducted interviews with developers who had practiced software interface design in diverse contexts, allowing additional insight into informal learning processes. To qualify for an interview, participants needed to (a) self-identify as having a computing background, (b) have designed an interface for at least one piece of software, and (c) self-identify as a novice-to-intermediate software interface designer. I recruited participants through mailing lists and forum announcements and provided incentives of $20 to those who completed the interview. Though I did not limit my recruitment to students only, all interview participants were current or recently graduated undergraduate and graduate computing students. Therefore, I refer to interviewees as students when reporting results.

Interview participants met with a researcher on University 2's campus for a semi-structured interview consisting of two main sections. First, participants relayed their background and education, including details about how they first learned to design software interfaces. Second, participants described the most recent piece of software for which they had designed or created a software interface, then walked the interviewer through their design process in as much detail as they could recall. After that, they discussed their "typical" design process (whether or not it aligned with their most recent project) and any particular challenges they recalled during design work.

I conducted 15 interviews in total. Participants described interface design work in a wide array of contexts, ranging from intensive year-long projects with external clients such as graduate capstones, to entirely on-the-job self-teaching of design principles and concepts, providing rich insights into varied learning challenges. I audio-recorded and transcribed the interviews, and took handwritten notes during the interviews to provide context. In total, I collected and analyzed about 245 minutes of audio.

Qualitative Analysis

Two researchers performed the qualitative analyses:

- The dissertation author, a computing education researcher with five years of research experience in HCI and design methods at the time of the study, including two years researching the overlap of software interface design and computing education.

- A research assistant with one year of research experience in computing education and design methods as well as one year of UX design experience.

First, the two researchers collaboratively affinity diagrammed the 117 survey responses to generate initial themes for the coding effort with a sensitizing concept of *types of learning challenge* [235]. Through iterative refinement and discussion, the researchers identified 13 types of challenge, which formed the basis of the code set used in subsequent analysis.

Next, the researchers performed two rounds of deductive qualitative coding on the transcribed interview data, segmented by sentence for analysis. In the first round of coding, to scope the amount of data to analyze, the researchers marked each sentence in the interview transcript as containing (1–N) or not containing (0) evidence to suggest the presence of a learning challenge. Sentences could contain multiple types of learning challenges, in which case researchers marked the number of distinct types present. The researchers conducted the first round of coding collaboratively over three 1-hour meetings, discussing interpretations and eventually achieving agreement.

The second round of deductive coding focused on sentences that contained at least one type of challenge. The researchers divided the data set and each qualitatively coded half the data using the code set of learning challenges identified in the surveys as a basis. I allowed for multiple codes per sentence (since one long sentence might contain evidence of many types of challenges) and no codes per sentence (since an interview participant might talk about a new type of challenge not observed in the survey data). Once the researchers finished their respective deductive analyses, they met to discuss interpretations and address any discrepancies in the application of the code set, then adjusted their coded data as needed.

Finally, for sentences in the interview data that appeared to contain challenges but did not fit into the existing code set, the researchers collaboratively affinity diagrammed and inductively coded the sentences to identify themes. This inductive analysis identified two additional types of learning challenges that were not present in the survey data. Adding these two new challenges produced an updated set of 15 total student-reported learning challenges, encompassing data from both the surveys and the interviews.

4.3.2 Study 1 Results: Student-Reported Challenges

Table 4.1 describes the 15 student-reported challenges I found during my analysis. I adhere to the perspective on qualitative coding presented by Hammer and Berland [132], treating the results of the coding effort as organizations of claims about data rather than quantitative data (i.e., measuring inter-rater reliability) in and of themselves. As a result, I do not report code frequencies, preferring instead to focus on descriptions of code instances observed in the data. For ease of reference, I assign each learning challenge observed in Study 1 a tag by which I refer to it throughout the rest of the chapter (see Table 4.1). I illustrate each challenge by providing two representative quotes: One from the survey data and one from the interview transcripts (or two from the interviews, if the challenge was not observed in surveys). IDs preceding survey quotes (e.g. U2C) represent the university and class of the quoted student. IDs preceding interview quotes represent the speaker (e.g. P8). I then provide a short description of each challenge by characterizing common themes that represented it in surveys and interviews.

Tag	Student Learning Challenge
`WHAT`	What is design?
`WHY`	Why do we do this design activity in this way?
`HOW`	How do I perform this design method?
`INFO`	How/where do I find a design resource?
`ADAPT`	How do I adapt parts of this design into my design?
`SYNTH`	How do I interpret this feedback?
`TEAM`	How do I work with my teammates effectively?
`STAKE`	How do I work with clients and stakeholders effectively?
`LIMIT`	How do I design with limited resources?
`SCOPE`	How do I scope this design problem?
`STAGE`	When should I move to the next design stage?
`EVAL`	How can I choose between options?
`BIAS`	How can I avoid biasing my design?
`DIVRS`	How do I design for diversity?
`ID`	Am I the kind of person that can or should do design?

Table 4.1: Descriptions of student-reported learning challenges identified in Study 1 surveys and interviews.

WHAT: What is design?

- U2C:*“I’m confused about what exactly counts as a prototype? Does it have to be a physical object?”*
- P8:*“[Researcher asks how they first learned design.] I did not. [laughs] I just did it. ... I don’t think I’ve actually ever been told how things should be or how things should look... I have no knowledge of how I *should* design things.”*

WHAT challenges occurred when a student lacked declarative knowledge, such as facts about design. Students reporting these kinds of learning challenges were confused about the nature of various design objects (e.g. prototypes, wireframes) and activities (e.g. stakeholder analysis, sketching). In the surveys, though some WHAT challenges should be expected with novice designers, students still reported uncertainty about the nature of design concepts *directly after* lessons on that topic, as the U2C student did above after a lesson on prototyping. In interviews, students reported WHAT challenges when they discussed having to do design work without much (if any) formal training. Recall that an inclusion criteria for the interviews was that the participant had designed at least one software interface; despite this, some of the interviewees reported that they did not even know what UX design was when they began their projects, or that they never had a concrete design process.

WHY: Why do we do this design activity in this way?

- U2C:*“The thing I made on prototype won’t show on the actual app. What’s [the] point for us to prototype unuseful interface?”*
- P5:*“So unfortunately user experience was the last part of it. ... We started off by making it a very useful tool. And usable, but then usable came second.”*

WHY challenges arose when students did not understand the reasoning behind performing a design activity in a particular way. In the surveys, these challenges manifested as questions about why students were spending time working on interfaces (prototypes) that were not the actual end products, as well as questions about the reasoning behind particular design methods’ utilities (such as brainstorming in isolation before bringing your ideas to a large group). `WHY` challenges did not necessarily prevent students from practicing design: Students reporting them in interviews still completed design work, though they simultaneously reported confusion about the rationale behind design tasks. This led some students to question the importance of design work overall (such as P5 described above) and put off interface design work until the final stages of implementation.

`HOW`: How do I perform this design method?

- U2B:*"How do we deal with having more than one design in the beginning? ... How do you compare ideas?"*
- P7:*" [We used] trial and error... My mentor and I would have an idea, we'd implement it, and we'd test it with people. And then it would crash and burn and work terribly."*

Students who reported `HOW` challenges in the surveys asked questions about how to perform the steps of different design methods. In the interviews, students like P7 often knew that their current practices were not necessarily optimal, but they did not know what steps they could take toward formal design methods. Interview participants that indicated these types of challenges sometimes went on to describe how confusing their (lack of a) design process was, or how the resulting design was poor quality and took longer to finalize than expected.

`INFO`: How/where do I find a design resource?

- P14:*"When you learn new software, you don't know what the software is capable of doing. [P14 describes how they would follow YouTube tutorials when available.] But there's sometimes things that I cannot look for [with] tutorials. There's no tutorial online about the topic that I want to go in. ... all little skills that I needed to pick up because someone didn't teach me that."*
- P12:*"It was just us developers trying to do as well as we can. I hadn't studied the UX process before. ... While I was working, I just would consume as many articles as I could on the web. But there was no process I could follow."*

I only observed `INFO` challenges in the interview data. These kinds of challenges most often manifested as students sought information to help them design – whether in the form of a tutorial, a description of a design method, an article about interface design, or an example to inform their design. Interview participants who indicated they struggled with `INFO` challenges sometimes recalled that issues finding resources slowed their progress or prevented them from doing design work altogether. `INFO` challenges often came up when students had to learn design concepts independently (i.e. that their coursework had not taught them).

`ADAPT`*: How do I adapt parts of this design into my design?*

- P11:*"I found some websites for [design] references in different systems. But, it's like, this looks pretty, but how do I apply it to my prototype? I don't know how to do that."*
- P14:*" It's better for me to look for how people do it because when they're doing it, I can learn about other stuff as well as like how the system works and how do they come together to achieve the thing that I want to do... Later when I start to get used to the software, I would change a little bit. It's like, oh, okay, here they do [a transition] like 0.5 second, I can do like 0.3. ... Start doing those little changes that is fitting my expectation more."*

`ADAPT` challenges also only appeared in the interview data. These challenges revolved around students' struggles to adapt elements of an example to their own designs. "Stealing" successful solutions to analogous design problems [118, 134] and composing features of those ideas to create novel solutions [216, 289] are both known design proficiencies, but computing students reporting `ADAPT` challenges struggled to perform these tasks. Some students who reported this kind of challenge were on teams consisting only of software developers (no designers) and felt they did not have anyone to turn to for assistance. In any case, students experiencing these challenges often took longer than they expected to finish design tasks due to the extra time needed to adapt designs.

`SYNTH`*: How do I interpret this feedback?*

- U1A:*"How do I prioritize the information/research I've completed so that I can properly inform* *my design?"*

- P9:*"Our findings from research were in a different format than the design itself. So it's going from that stuff [results], like, we were doing an affinity analysis of findings from the research, from that to the visual layout and stuff like that. ... [You] kind of have to double check to make sure that you're actually doing the research justice and you're actually serving their needs in this new interface [version]."*

Students who reported `SYNTH` challenges often struggled to synthesize feedback they received from critique sessions or user evaluations in ways that could inform subsequent design choices. In the surveys, students who experienced this challenge often asked about how to derive requirements from broad initial research efforts, as well as how to determine the severity of usability issues discovered through testing. In interviews, students who struggled with `SYNTH` challenges spoke about uncertainty over whether their interpretations of feedback were "correct," especially when the feedback received did not correspond to an obvious design decision. Students also reported interpreting feedback incorrectly (i.e. in ways that did not make the interface more usable or useful), which required extra time and resources to remedy the error.

`TEAM`*: How do I work with my teammates effectively?*

- U2C:*"How to efficiently communicate with team members?"*
- P15:*"The initial developer I worked with, from the [country name] team, he had come with this fixed mindset about how much effort he needed to put in, not how much effort was actually required for the project. It was more like, 'Oh, okay, maybe we do need to do all this [work], but I'm only going to put in this much amount of time."'*

Collaboration with teammates on design work was often difficult for students. Students reported `TEAM` challenges effectively communicating ideas to others, resolving conflicts (such as over differing interpretations of results or ownership of ideas), or working alongside teammates who did not want to put in sufficient time to do design work well. Students reported that `TEAM` challenges slowed their progress, and a few students said they wished they had received training on how to effectively design in teams.

`STAKE`: *How do I work with clients/stakeholders effectively?*

- U2C:*"Is it true that clients don't know what they want until they have seen a lot of things they don't want?"*
- P9:*"The biggest issue that jumps to mind is the jargon that we [designers] use. ... Eventually, if you're in these environments, you just use it to describe normal situations. And then once you're with users... You have to kind of catch yourself, when you're describing how something is going to be used or how you're going to collect information ... explaining these concepts to non-technical users."*

Students also reported difficulties collaborating with clients and stakeholders on design projects. `STAKE` challenges manifested when students struggled to elicit requirements, communicate domain-specific information, or present results of design work to clients who lacked design domain expertise. In some cases, such as P9's above, students had to adapt their communication styles around clients and stakeholders, which many found difficult. In other cases, `STAKE` challenges arose when clients collaborated poorly with students, which prevented the students from gaining access to needed resources.

`LIMIT`: *How do I design with limited resources?*

- U2C:*"How to balance creativity and do-ability?"*
- P5:*"We started working on it [the design] and realized that maybe we should have asked them [users] that question, and then that becomes the second iteration, which is costly in terms of time and money. ... You don't want to go in iterations. You want to actually spend time to get lesser iterations of things. Ask the right questions the first time."*

Students often reported difficulties managing limited resources—whether the resource in question was time, money, access to users, or other constraints imposed by the environment they were working in. In surveys, students questioned how to prioritize design tasks based on cost-effectiveness, time-efficiency, and feasibility of implementation. In the interviews, students reported struggling with tight deadlines (especially those that forced them to change their planned designs), accessing representative users, and balancing practical concerns with the desire to meet all design goals. Students who experienced `LIMIT` challenges sometimes had to skip parts of the design process or otherwise leave out features.

`SCOPE`: How do I scope this design problem?

- U1A: *"How do you define a design problem?"*
- P9: *"Sometimes there is changing expectations ... There's certain things that we just couldn't do anymore ... scope, basically. Challenging to figure out, if we have one intention and that ends up not being feasible, how do we still honor the users, the expectations and requirements, while having to compromise on other parts of the application."*

Students reported `SCOPE` challenges when they tried to define the boundaries of design problems. These attempts at scoping sometimes occured at the beginning of the design process, when students first began to decide on what they wanted to create. `SCOPE` challenges also occurred in the middle of students' design processes as students realized that their initial conception of the problem was not adequate—a known byproduct of the design process in literature (c.f. "productive failure" as described in [239]). Students who struggled with `SCOPE` challenges reported delays in getting started or having to do "extra" unwanted iteration.

`STAGE`: When should I move to the next design stage?

- U2B: *"How many ideas is too many [during ideation]? When do you know you have enough?"*
- P5: *"It [the project]'s still not done yet, and I don't think there will ever be a point. I think we're going to keep taking suggestions. For me personally, I don't think we ever reach a point where everybody becomes happy."*

Students often reported `STAGE` challenges in the middle of their design processes, especially when trying to determine criteria which signified the need to move on with design work. Students wanted to know what was "good enough" to move on from brainstorming, how many user studies or usability tests needed to be run, or what constituted enough prototype iterations created. Those who experienced `STAGE` challenges reported frustration with design work that never seemed to be "done" (similar to P5 above) and confusion over when design requirements were adequately met.

`EVAL`: How can I choose between options?

- U2B: *"How do you choose the right method for the job?"*
- P10: *"We actually conduct different interviews with different user groups. Then we understand each groups' pain points, then kinda have to try to solve all the pain points at the same time. But of course, it's really hard. This is why sometimes you have to do some tradeoffs. ... For this one [project] we only focus on one of the user groups."*

`EVAL` challenges arose when students struggled to evaluate which of several options was most fit for their project. In the surveys, students asked questions about deciding on the "best" or "ideal" design method or activity, as well as how to decide between multiple applicable approaches. In interviews, students who reported `EVAL` challenges spoke about the difficulty inherent in evaluating tradeoffs when trying to satisfy the requirements of competing design goals and deciding which user needs to prioritize over others. Students who did not know how to progress past `EVAL` challenges sometimes reported spending more time than they would have liked in planning stages, or choosing arbitrarily between options without grounding the decision in design rationale.

`BIAS`: How can I avoid biasing my design?

- U1A: *"How do we know if a project is truely [sic] a usability issue or if we are displaying confirmation bias?"*
- P8: *"I don't think I've actually ever been told how things should be or how things should look ... I just judge based on things I've seen that I like. I kind of evaluate everything as *I* look at it.*

But I have no idea how other people will interact with it."

Students who reported `BIAS` challenges struggled to prevent their own inclinations and biases from impacting their design decisions. In the surveys, students often reported being aware that their biases could influence their designs (as in the above survey response), but did not know what to do about it or how to recognize when it had happened. In the interviews, students experienced `BIAS` challenges when they assumed that if they personally found the design aesthetically pleasing and usable, others surely would as well. Sometimes, like P8 described above, students did not know any better ways to design interfaces than simply relying on their own evaluations. Students who reported these challenges often later (e.g. during user testing) found that their designs were not as high quality as they had believed, leading some to start over completely.

`DIVRS`*: How do I design for diversity?*

- U2C:*"How do you design an application that works for everyone?"*
- P1:*"[When designing] I'm thinking about how do I build something for the people that I don't understand, necessarily, their cultural experiences or how they view, or their perception of something. ... It [the application] was supposed to be specific to [region other than designer's place of residence], but you know, when it comes to development stuff, there's just so, so many differences between culture and that kind of thing that it still makes it difficult."*

`DIVRS` challenges imply that students struggled to perspective-take or empathize with their interface's target users. Students reported challenges designing for diverse abilities and usage styles, especially when users' experiences were very dissimilar to their own. Challenges around `DIVRS` often came up alongside designing for accessibility or inclusion, but were part of broader discussions around usability as well, especially when students began to realize that people could interpret designs quite differently. Students who reported `DIVRS` challenges in the interviews sometimes went on to describe how their design solutions failed to represent users' true needs, and were therefore less useful than intended.

ID: Am I the kind of person that can/should do design?

• U2C:*"Is design an easy job that every one can do?"*

• P7:*"I'm a person who does design to fill the need of there being design done. ... If I compare myself to people who identify as UX designers, I think they spend a lot more time with wireframes and paper prototypes and thinking about the theory behind their designs. I don't really identify as a UX designer, I'm just a person who designs things. I build stuff and test it, and if it doesn't work I change it."*

ID challenges were one of the least commonly observed in my data, though potentially some of the most problematic. These kinds of learning challenges manifested in the surveys when students asked about intrinsic qualities designers should possess to be successful. In the interviews, students' ID challenges sometimes manifested when students were reluctant to claim the title of designer, even when they clearly performed design tasks, like P7 above. Students reporting this reluctance spoke about designing out of necessity rather than choice (e.g., they were working on a developer team with no designers) and not feeling like they actually "did" design work, even if they had clearly made design decisions and performed design activities. ID challenges may be tied to a lack of design self-efficacy [13]: Students may not have been confident in their design abilities, and thus chose not to identify as designers.

4.4 Study 2: Educator Perspectives

The student perspectives represented in the previous section are important to understand HCI education learning challenges, especially as the 15 challenges I found were observed across multiple learning contexts. However, educators can also provide perspectives on learning challenges in the HCI classroom, which can consist of multiple years of experience watching their students struggle to learn software interface design concepts. To learn from these experiences, and to further validate the existence of the student challenges presented above, I designed and deployed an online survey.

4.4.1 Method

Survey Structure

I created and deployed the English-only educator survey using an online survey platform. The survey began by verifying that the educator met inclusion criteria: (a) 18 years of age or older, (b) taught computing students (here, presented as "students who may create software interfaces in their future careers" to signal inclusion of non-CS departments), and (c) taught software interface design concepts to these students. The survey took educators who affirmed all three to the next set of questions, which I designed to validate the existence of the 15 student-reported challenges uncovered by Study 1.

For each of the 15 challenges, I presented educators with a description of the challenge type (similar to the text in Table 4.1), then asked them to report if they had observed this type of challenge in their classes. Educators could respond in three ways: "Yes", "No, but I believe students might experience this challenge", or "No, and I don't believe this challenge exists." The two "No" variants added a small amount of descriptive data to an otherwise closed-ended survey response and allowed me to better understand educators' perceptions of these challenges. I held the order of items corresponding to `WHAT` and `WHY` challenges constant across surveys to allow educators to acclimate to the question format, since I believed these two kinds of challenges to be easily identifiable to educators. The subsequent order of items corresponding to the other challenges was randomized to limit fatigue effects.

To supplement the above closed-ended survey responses with qualitative data, I included one open-ended item asking educators to describe an interesting instance of student learning challenge they had observed. I hoped for this item to surface learning challenges I had not observed in Study 1. Finally, I ended the survey with demographic questions about educators' backgrounds and a field for educators to leave their email if they were open to a follow-up interview.

Recruiting & Respondents

I recruited through four channels, offering a high-level summary of the survey's responses and implications for teaching as an incentive:

- Twitter. My tweet received 7,204 impressions, 18 retweets, and 21 likes, and 33 clicks on the link by survey close.
- A closed Facebook HCI educator group with 217 members. The post was seen by 80 members and received 3 likes.
- Two Slack groups (40 members and 54 members) targeted at HCI educators, whose membership likely overlapped significantly with the Facebook group due to shared leadership.
- Targeted emails (77 total) to HCI and interface design educators who provided their contact information to the research team during previous studies relating to this topic.

The survey remained open for 26 days, with the majority of responses received in the 1st week. I received 52 responses to the survey. Of those 52, 36 finished the entire survey (a drop out rate of 30.8%), and of those 36, 35 (97.2%) met my inclusion criteria. I discarded the 17 responses that were unfinished or did not meet inclusion criteria. The results below therefore represent perspectives from 35 HCI educators who teach software interface design concepts to computing students. To put the 35 responses in perspective, in recent proceedings, a few hundred institutions publish HCI-related work at the ACM CHI conference each year. Assuming one educator who fits my target population at each institution, the 35 responses might represent 5-10% of the current population of HCI educators. While this number is still relatively small, I feel that it is representative enough to provide initial insights, especially since many educators reported similar themes in their open-ended responses (suggesting saturation).

Table 4.2 shows an overview of educators' demographics. Most taught in the United States at large, public universities, and most self-reported their main field of study or practice as HCI or CS. The educators reported a wide range of years of teaching experience, though most reported 1-5

Country		Main Field		Years Teaching (Total)		Years Teaching HCI		Main Teaching Institution	
US:	26	HCI:	16	<1 year:	0	<1 year:	2	Large, public university:	21
Canada:	2	CS:	9	1-5 years:	15	1-5 years:	15	Small, public university:	5
Germany:	2	Soft. Eng:	3	5-10 years:	5	5-10 years:	5	Large, private university:	3
Austria:	1	Design:	3	10-20 years:	6	10-20 years:	8	Small, private university:	2
Denmark:	1	CSCW:	1	20+ years:	9	20+ years:	4	Community/Junior College:	1
Morocco:	1	Web Design:	1					Professional training program:	1
Philippines:	1	User Research:	1					Other (did not report):	1
UK:	1	UX:	1						

Table 4.2: Demographics of the 35 educators who responded to the Study 2 survey.

years of overall experience and 1-5 years of experience specifically teaching interface design skills to computing students.

Follow-up Interviews

To further explore educators' perspectives, I followed up with a subset of the educators who both left their email and answered the open-ended question about an interesting instance of student learning challenges. The goals of these follow-up emails included clarifying details of educators' responses and gaining insight into educators' perceptions of the student learning challenges. Of the 35 educators, 17 provided both contact information and an description of a time they noticed students struggle. I reached out to 13 of these educators by email with targeted questions about their open ended responses. Of these, 8 responded, providing additional qualitative data.

Qualitative Analysis

I combined the data received from educators' follow-up interviews with the data from the open-ended responses on the survey. The resulting data consisted of a set of qualitative descriptions of student learning challenges, encompassing perspectives from 27 of the 35 educators. To analyze this data, the same pair of researchers from Study 1 performed a thematic analysis [235]. The primary analyst (the dissertation author) examined the text of the qualitative data and annotated it with memos indicating each time an educator wrote about a type of student learning challenge. The secondary analyst (the research assistant) did the same, verifying the primary analyst's notations and adding their own. The two analysts then collaboratively affinity diagrammed the memoized data with a sensitizing concept of *types of learning challenge* to align with the analysis perspective used in Study 1. Loosely, this resulted in two categories of data: Student learning challenges that I identified in Study 1 (which served to verify the existence of student-reported challenges), and new challenges that I had not observed in Study 1. For the data that indicated new challenges, the analysts performed a subsequent round of collaborative inductive coding to surface 3 new types of student learning challenges, which I present below. As before, I do not treat the results of this analysis as quantitative data, but rather as an organization of claims about data [132].

4.4.2 Study 2 Results: Educator-Reported Challenges

Table 4.3 shows the results for the closed-ended survey questions about the student-reported challenges from Study 1. For each of the 15 student-reported learning challenges, at least some educators reported they had observed it in their classes. Educators' open-ended responses also described instances of nearly all of the struggles that students had self-reported in Study 1 (see Table 4.4 in the Discussion for an overview).

My qualitative analysis of educators' open-ended responses discovered three additional learning challenges beyond those I discovered in Study 1. The overarching theme tying these three challenges together was that *students did not necessarily perceive challenges they experience as struggles*, even though, from an educator's perspective, it was clear that the student was not successfully learning or applying design knowledge. One educator characterized these challenges as follows:

- *"Often the problems I see are best categorized as "unknown unknowns"—where the student confidently conclude[s] they know what to do next, how to ask a question, or how to apply a design principle (or decide they don't need to apply it), but are actually wrong."*

	Educator Responses (out of 35)		
Challenge	**Yes, seen it**	**No, but might exist**	**No, does not exist**
`WHAT`	19 (54.3%)	14 (40.0%)	2 (5.7%)
`WHY`	22 (62.9%)	13 (37.1%)	0 (0.0%)
`HOW`	21 (60.0%)	13 (37.1%)	1 (2.9%)
`INFO`	15 (42.9%)	14 (40.0%)	6 (17.1%)
`ADAPT`	13 (37.1%)	17 (48.6%)	5 (14.3%)
`SYNTH`	23 (65.7%)	11 (31.4%)	1 (2.9%)
`TEAM`	31 (88.6%)	4 (11.4%)	0 (0.0%)
`STAKE`	18 (51.4%)	15 (42.9%)	2 (5.7%)
`LIMIT`	24 (68.6%)	10 (28.6%)	1 (2.9%)
`SCOPE`	30 (85.7%)	5 (14.3%)	0 (0.0%)
`STAGE`	25 (71.4%)	9 (25.7%)	1 (2.9%)
`EVAL`	21 (60.0%)	13 (37.1%)	1 (2.9%)
`BIAS`	17 (48.6%)	17 (48.6%)	1 (2.9%)
`DIVRS`	19 (54.3%)	13 (37.1%)	3 (8.6%)
`ID`	14 (40.0%)	18 (51.4%)	3 (8.6%)

Table 4.3: Frequency of educator responses on the Study 2 survey for items corresponding to student-reported learning challenges from Study 1. Percentages indicate proportions out of 35.

`WARP`: *Students hold inaccurate perceptions of design*

Some educators reported that their students held inaccurate perceptions of what design entailed or how it related to technical (programming) work. For instance, one educator described how making the interface design class mandatory for software engineering majors revealed resistance to learning:

- *"[M]any of the students actually had little to no interest in engaging with the material and often had condescending comments such as "I don't get the point of all this requirements gathering"... definitely was a challenge to explain to a lot of these students why design thinking mattered."*

Other educators reported similar resistance in their classes, relaying that students who thought interface design was only about making the software "look pretty" sometimes failed to engage with class material enough to learn anything. One educator tied `WARP` challenges to design self-efficacy:

- *"A lot of students have been conditioned to think that they "can't" do certain things (e.g. drawing), and it's really hard to get them out of the mindset. It sometimes turned into stubbornness, where a small number of students have tried to "prove" they don't need interaction design to do things and they know better."*

These challenges are consistent with prior work in HCI education reporting inaccurate perceptions of interface design from computing students [55, 122].

`STUCK`: *Students fixate on conventional design patterns*

Educators also reported that students often adopted elements of conventional designs without considering if these elements fit their specific design goals, assuming that there were certain aspects of interfaces that were "not allowed" to be changed:

- *"[S]tudents struggle most with thinking deeply about the root cause of usability issues and rethinking bigger decisions... Students are often most comfortable adopting what they see as a standard design or approach and have a harder time rethinking fundamental assumptions* *[that] they never considered to be explicit choices at all.”*

One educator elaborated that STUCK challenges prevented students from designing software that fit their users’ needs:

- *“In many cases it seems they had a solution in mind and focussed [sic] on this solution rather than finding out more about the participants.”*

Design fixation is a known problem for novice designers, who may not even be aware that they are fixating [153].

RUSH: Students rush to implement and discount design work

Finally, educators reported that students often rushed through the early stages of design work and focused entirely too much on implementation details. One educator reported their students rushed through prototyping:

- *“Many students like to jump into creating a higher-fidelity prototype from the beginning. They struggle to justify why it is important to start implementing their design ideas through low-fidelity prototyping.”*

Another reported that RUSH challenges might lead students to focus on low-level details before solidifying the high-level structure of their designs:

- *“Confusion between wireframing and high fidelity mockups. Students might spend time on visual design while still in the ideation/architecture stage.”*

One educator related RUSH challenges to the way prior classes conditioned students to approach programming problems:

- *“They tend to approach interface design like programming in that they assume that if they do the steps and get some results, then they are successful. It’s something of a “as long as it compiles and runs on the test data, my job is done” mentality. I find the most success when I (or my TAs) push them to consider many of the issues you brought up [in the survey]; otherwise they will just get things done as quickly as possible, a bad recipe for interface design.”*

Educators reported that students who struggled with RUSH challenges produced designs that provided little value to their users, though students often failed to identify this behavior as the cause

		Students (Study 1)		Educators (Study 2)		
ID	**Description**	**Surveys**	**Interviews**	**Surveys**	**Qual. Data**	**Prior Work**
`WHAT`	What is design?	✓	✓	✓	✓	[29]
`WHY`	Why do we do this design activity in this way?	✓	✓	✓	✓	[29]
`HOW`	How do I perform this design method?	✓	✓	✓	✓	[147]
`INFO`	How/where do I find a design resource?		✓	✓		[241]
`ADAPT`	How do I adapt parts of this design into my design?		✓	✓	✓	[118, 289]
`SYNTH`	How do I interpret this feedback?	✓	✓	✓	✓	
`TEAM`	How do I work with my teammates effectively?	✓	✓	✓	✓	[53]
`STAKE`	How do I work with clients and stakeholders effectively?	✓	✓	✓	✓	[53]
`LIMIT`	How do I design with limited resources?	✓	✓	✓	✓	
`SCOPE`	How do I scope this design problem?	✓	✓	✓	✓	[1, 115]
`STAGE`	When should I move to the next design stage?	✓	✓	✓	✓	
`EVAL`	How can I choose between options?	✓	✓	✓	✓	
`BIAS`	How can I avoid biasing my design?	✓	✓	✓	✓	[228]
`DIVRS`	How do I design for diversity?	✓	✓	✓	✓	[228]
`ID`	Am I the kind of person that can or should do design?	✓	✓	✓	✓	[13, 16]
`WARP`	Students hold inaccurate perceptions of design.				✓	[55, 122]
`STUCK`	Students fixate on conventional design patterns.				✓	[75, 153]
`RUSH`	Students rush to implement and discount design work.				✓	

Table 4.4: Triangulation: Each student-reported learning challenge was supported by at least three data sources, while the three educator-reported learning challenges indicate struggles students might not have known they faced.

of their poor results.

4.5 Discussion and Concluding Remarks

The goal of this study was to identify different kinds of challenges computing students face when learning about software interface design in order to support the development of HCI pedagogy. Table 4.4 lists each type of challenge I observed and the data sources supporting it, including relevant ties to prior work. I found at least four overarching categories of challenges reported by students and educators:

- *Challenges around how to do design work (`WHAT`, `WHY`, `HOW`, `INFO`, `ADAPT`, and `SYNTH`).* These arose when students struggled to understand the mechanics of interface design work, and often slowed down or prevented students' progress on design problems.

- *Challenges around project management skills (`TEAM`, `STAKE`, and `LIMIT`).* These arose when students struggled to collaborate with others or manage limited resources, sometimes leading to communication breakdowns or the abandonment of parts of the design process.

- *Challenges around the wickedness of design problems (`SCOPE`, `STAGE`, and `EVAL`).* These arose when students struggled with the "wickedness" [249] of design problems with unclear definitions and no definitively correct answers. Students facing these challenges reported frustration and confusion over the ambiguity of design work.

- *Challenges around distorted perspectives (`BIAS`, `DIVRS`, `ID`, `WARP`, `STUCK`, and `RUSH`).* These arose when students either had difficulties taking the perspectives of others, or when they did not realize that their own perspectives were at odds with designing high quality interfaces. Students may or may not have realized they faced these challenges.

The set of 18 student learning challenges presented in this chapter provides one component of the knowledge needed to more effectively teach software interface design concepts to computing students. For some of the challenges (`WHAT`, `WHY`, `HOW`, `INFO`, `ADAPT`, `TEAM`, `STAKE`, `SCOPE`, `BIAS`, `DIVRS`, `ID`, `WARP`, and `STUCK`), prior work from learning science, HCI, software engineering, or design education indicates that they might be difficult for students who are novice designers (see Table 4.4). Others (`SYNTH`, `LIMIT`, `STAGE`, `EVAL`, and `RUSH`) appear to be undiscussed in relevant prior literature, which may imply that they are unique to this topic and audience.

Though the data I collected was rich, some aspects of my study design limit the generalizability of these findings. Due to the high variation between HCI courses across institutions, I cannot be sure that these observations generalize across all contexts. For Study 1, my surveys gathered data from students at only one single instant during instruction and were presented slightly differently to fit the context of each class. The surveys also were only deployed at two U.S. based universities.

My Study 1 interviews were conducted in-person on a university campus, which may have limited participation. Further, students in Study 1 likely varied in their ability to reflect on their own learning. The educators' perspectives provided in Study 2 expanded my understandings of student learning challenges, but they also came from a relatively small number of educators who fit my inclusion criteria. Several factors likely influenced what kind of data I was able to collect, such as the timing of the survey's deployment, the kind of educators who were motivated enough to answer my survey, and educators' own abilities to reflect upon and recall students' experiences in their classes. To safeguard against these limitations, I relied on extensive use of triangulation with multiple data sources and with prior work, as seen in Table 4.4. However, some of the interpretations I present might have been different if I had studied other students or other teachers. Future work in this area should attempt to discover if these challenges persist across varied educational contexts and whether other challenges exist that I did not observe.

Nonetheless, my findings reveal a number of interesting implications for research. For instance, how prevalent are these challenges in broader contexts? Under what conditions (e.g., studio-based vs. traditional lecture-based classes) might computing students experience these kinds of challenges more or less often? As HCI expands beyond higher education into primary and secondary curricula (like Exploring Computer Science [120] or Code.org [59]), will these learning challenges still hold? And what are effective strategies to mitigate students' learning challenges that fit these categories? The `RUSH` challenge revealed by educators in Study 2 also suggests an interesting hypothesis: the way we teach computing students to create software and write code may make them less likely to succeed at interface design work. Future work in this area should explore the extent to which prior computing knowledge influences students' experiences with these challenges.

My results also contribute to the discourse around *pedagogical content knowledge (PCK)* [266] development for HCI design education. PCK is domain-specific [128, 146, 150] and consists of knowledge of pedagogical strategies to teach a particular topic, in a particular context, to a particular audience. Exact definitions of the components of PCK vary (c.f. [29, 108, 211]), but knowledge of student learning challenges is generally considered a core aspect. Our field has only begun to investigate the nature of computing PCK within the past decade, from primary and secondary learning environments [27, 112, 207, 243, 292], to both general [145, 146] and specific [163, 169, 185, 194, 225, 228, 312] aspects of post-secondary CS education. A prior study of ours did explore PCK for teaching software interface design skills [228], but it was scoped specifically to teaching a particular gender-inclusive interface design method and focused on educators' pedagogical strategies rather than students' perspectives. Therefore, the set of student learning challenges described in this chapter provides some of the first foundations for future research on PCK for general HCI design education. Further exploring this space might enable more effective use of instruction time in HCI classes (which are known to suffer from time constraints already [55]) through the development of more effective learning materials, or even help shorten the onboarding time for new HCI design educators—an important pursuit to ensure we have enough teachers to keep pace with the rapid growth of computing education.

Equipped with this better understanding of student learning challenges, we can begin to deepen our understanding of how to provide computing students with effective design educations. Implementing this newly gained knowledge in curricula and pedagogy will lead to better teaching and learning around HCI design concepts. Through this effort, the software industry as a whole will benefit from a pool of design-literate computing graduates who enter the workforce ready to understand and contribute to many aspects of large projects, aware of the impacts of their design choices. Developers will be empowered to design usable, accessible, ethical, and inclusive software interfaces, allowing more diverse populations to engage with various technologies and participate in today's computing-infused world.

Chapter 5

TEACHING INCLUSIVE DESIGN SKILLS WITH THE CIDER ASSUMPTION ELICITATION TECHNIQUE

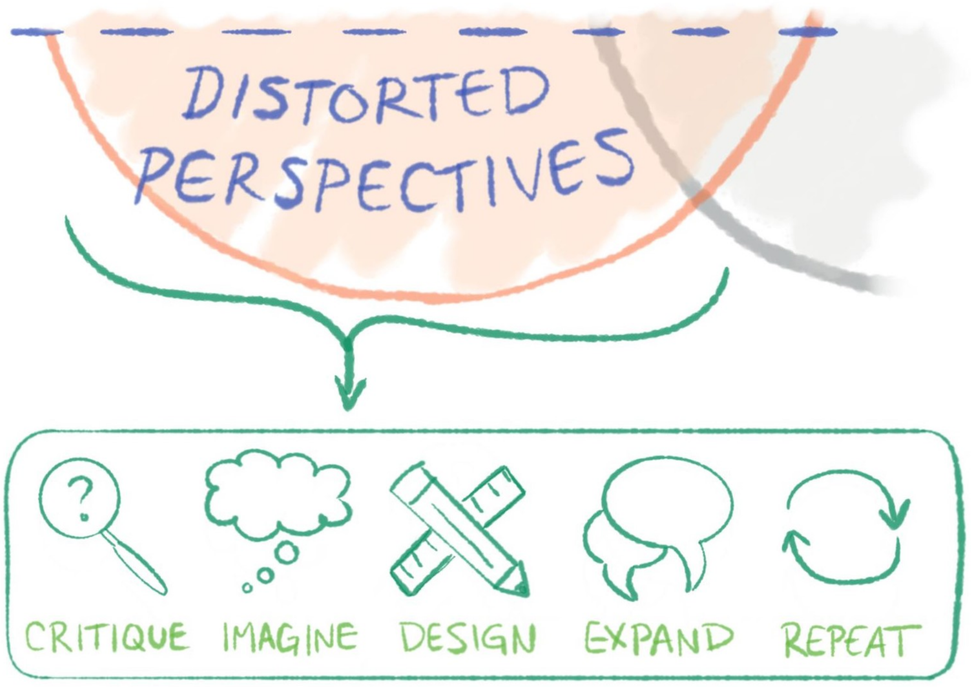


Figure 5.1: Informed by the “distorted perspectives” category of learning challenges from Chapter 4, the study described in this chapter contributes a five-stage technique called CIDER that used a strategy called assumption elicitation to teach inclusive design skills.

5.1 Introduction

In today's connected world, everyone should be able to effectively and equitably interact with technology so they can access the benefits it provides. Unfortunately, software and hardware interface designs often fail to support different kinds of users well. Prior work in the human-computer interaction (HCI) and user experience (UX) design spaces documents a plethora of cases in which a technology's design might discriminate against users of varying physical or mental capabilities [252, 279, 317], races or ethnicities [224, 245, 255], cultures [7], genders [57, 131, 209], and socioeconomic statuses [208], among other facets of diversity. These kinds of design exclusion often arise due to biases held by designers, who embed their values into the artifacts they create [101].

One prevalent type of design bias follows from the implicit or explicit assumptions designers make about their users. Reliance on some sort of assumptions is inevitable when designing because a single designer or design team cannot predict everything about how their design might be used [34, 189, 294], even if they leverage participatory or community-based design methods. However, exclusion often arises when the assumptions a design rests on involve users' capabilities, contexts, identities, or environments. Prior work describes how technology is often created with an imaginary "average user" in mind, and that this hypothetical user is usually of a socially or culturally dominant race, gender, age, culture, and class, who is heterosexual, affluent, comfortable with technology, and not disabled [70]. Designs made for "average" users tend to work well only for users from who fit all these categories. When design decisions are predicated on assumptions that users possess particular characteristics or have access to particular resources, the design of the resulting artifact often disproportionately disadvantages users from already-marginalized groups [307].

To counteract the harmful effects of design bias and contribute to more inclusive designs, those who design technology should be equipped with the skills to recognize and respond to harmful assumptions about users. Unfortunately, prior work from the area of HCI education documents a number of unresolved challenges people might face when learning or practicing inclusive design skills. Persistent problems reported include difficulties getting learners to recognize that they are making assumptions about users in the first place [192] and to consider diverse perspectives when designing [228]. Students may also lack the experience, knowledge, or perspective-taking skills to understand how their design choices can exclude different groups of users, or experience confusion about how seemingly abstract notions like inclusion manifest in actual design features [229]. Finally, students might actively trivialize inclusive design work, showing resistance to learning about such topics [228] or not taking the material seriously [229]. Because the way practitioners learn to design can impact their future practice [201, 203], successfully addressing these challenges in educational contexts may encourage more inclusive design approaches when current students transition into professional technology design practice.

Toward this end, I created the CIDER assumption elicitation technique, an educational analytical design evaluation method. CIDER, which stands for *Critique, Imagine, Design, Expand, Repeat*, aims to help designers recognize and respond to interface and interaction design bias in technological artifacts as a means of teaching inclusive design skills. In the following sections, I describe the theoretical grounding of CIDER, which draws on prior work from the areas of design rationale, perspective taking, and inclusive design education to help learners build concrete understandings of how inaccurate assumptions about users lead to design bias and exclusion. The novelty of this technique is twofold: To my knowledge, CIDER is the one of the first educational design techniques to explicitly use *assumptions about users* as its basis, tying together design features, design decisions, and the impacts designers' conceptions of users have on the inclusiveness of an artifact. In addition, the stages of the CIDER technique support learners in building concrete understandings of design bias and inclusion in actionable ways that they can apply to future design practice. This directly addresses unresolved challenges raised by recent HCI/UX design education work (e.g. [229]).

To evaluate the efficacy of the technique, I conducted a concurrent embedded mixed-method [72] case study in a post-secondary interaction design course where I explored the impact of CIDER on students' understandings of inclusion (n=40 students, with data collection spanning eleven weeks of instruction plus follow-up interviews one month later). This study investigated four research questions:

1. How might CIDER-based activities impact students' self-efficacy as a designer?

2. How might the CIDER technique help students recognize different types of exclusionary design biases?

3. How might conducting CIDER-based activities collaboratively, rather than individually, impact students' experiences?

4. What kinds of lasting impacts might the CIDER technique have on students' design approaches?

Finally, I discuss considerations for using the CIDER technique to teach inclusive design, as well as implications for my findings for future research on inclusive design pedagogy and educational practice. My contributions are: [1] [2]

1. The theoretically-grounded CIDER assumption elicitation technique for teaching inclusive design skills;

2. An example of how the CIDER technique could be used in post-secondary introductory design courses to promote inclusive technology design; and

3. Insights into how CIDER-based educational activities might promote better understandings of inclusion over time, with potential for lasting positive impacts on early designers' design approaches and attitudes toward inclusive design.

[1]Throughout this chapter, I use first-person language ("I", "my", etc.) to describe this work to reflect the single-author nature of the dissertation document. However, this work was conducted in collaboration with Meron Solomon, Christopher Perdriau, and Dr. Amy J. Ko, and published in in ACM Transactions on Computer-Human Interaction (TOCHI) Volume 30, Issue 1 in March 2023 [230]. I acknowledge the shared contributions of all authors to the original publication.

[2]This material is based upon work supported by the National Science Foundation under Grant no. 1539179, 1703304, 1836813, 2031265, 2100296, 2122950, 2137834, 2137312, DGE-1762114, and unrestricted gifts from Microsoft, Adobe, and Google.

5.2 Background & Related Work

The goal of this chapter is to motivate and describe the CIDER assumption elicitation technique, as well as to explore its effectiveness in helping design students recognize and respond to bias. My working definition of HCI/UX design in this chapter draws on Park and McKilligan's model [233] and includes design practices related to interface, interaction, and UX design for HCI artifacts. I use the term *HCI artifact* inclusively to refer to both software and hardware with computational or computing-related components. For the purposes of this chapter, I define an *assumption* within a design to be a way in which a design's features and affordances rely on the user to have particular capabilities, resources, means, or knowledge, without which they might find it difficult to interact with the artifact as intended by the designers[3]. I then define *design bias* to be the ways in which assumptions might make it disproportionately difficult for particular (groups of) users to interact as intended with an HCI artifact, and *inclusive design* broadly to be an approach to or action of design work that recognizes and attempts to mitigate design biases.

In the following subsections, I motivate the need for a technique to teach inclusive design skills, drawing on work from HCI and UX education to illustrate challenges to learning and teaching inclusive design that arise in educational contexts. I also describe some of the ways in which students might struggle with designing inclusively for diverse user groups, highlighting the need for design methods to help students resist stereotyping behavior. I then draw on literature from software design and engineering to describe the role of assumptions in design through the lens of design rationale, illustrating how making assumptions visible can lead to higher quality designs. Finally, I describe how existing design evaluation methods may be used to surface assumptions (though assumptions themselves are rarely an explicit focus of existing methods), situating CIDER in the existing gap of evaluation methods designed for educational contexts that help students build concrete and actionable inclusive design skills.

[3]For instance, two assumptions which might be embedded in the design of a QWERTY desktop computer keyboard are that the user has enough fine motor control to press small keys in a particular sequence, or that they can recognize Latin/Roman alphabet characters.

5.2.1 Teaching and Learning Inclusive Technology Design

Though HCI education is a relatively young discipline [56, 302], there is a quickly growing focus on teaching computing students to critically consider the implications of the technologies they design. Fiesler et al. give an overview of the kinds of topics that are taught as "tech ethics," noting that much of this work has been published only within the past few years [96]. Some strategies for teaching the broader impacts of technology design focus on critiquing algorithmic design biases [246]. Others focus on teaching accessibility principles in standalone design courses [198], embedded in programming courses [156], or integrated throughout computing curricula [293]. Still others propose to help designers better understand the (sometimes conflicting) perspectives and values of various stakeholders, sometimes through techniques like Wong and Nguyen's *Timelines* value advocacy activities [309] or Cooney's notion of *micro-exposures* [66].

Nonetheless, prior work often reveals challenges that arise when teaching and learning design skills in computing-centric contexts. In general, teaching design principles to computing students can be difficult due to the ways that best practices for design pedagogy (c.f. studio approaches [28, 206]; "correctness" on wicked problems [94, 268]) conflict with the kinds of well-defined problems students tackle under traditional computing pedagogy [229]. Educators may not feel they have the expertise [265], time [165], or organizational supports [165] they need to properly teach accessible or inclusive design.

Students might also hold misconceptions about what design is or what it entails, leading them to devalue its importance. For instance, students can erroneously believe that design is strictly about aesthetics [229], that design work lacks rigor [55, 130], or that good design is just common sense [55]. These beliefs may be more prevalent among students with more technical backgrounds [193, 229], such as those in post-secondary HCI courses within computer science or information science programs. If these ideas are left unchallenged, students may have difficulty motivating themselves to address issues of inclusive design, implicitly assuming that these issues do not actually exist in in real-world designs or, at least, that they are not as critical as they are portrayed to be.

Sometimes, students may already value inclusion and inclusive design in an abstract sense, but they do not know how to translate those values into actual design decisions and actions, whether due to a lack of mechanical design knowledge [73] or of what inclusion means in terms of design [229]. Further, students with technical backgrounds can struggle with creativity when designing [75, 302] and fixate on what they consider to be design "standards" [219, 229], regardless of whether these designs actually fit users' needs. Prior work also suggests that the contexts in which early designers learn and practice design skills can influence their design values: If they are not supported, they might lose existing desires to prioritize inclusion in their designs [125, 295]. To ensure that learners who already value inclusion can translate their values into design practice, they need explicit support and scaffolding [54]. Otherwise, they may give up on designing inclusively due to confusion or a perceived lack of return on their efforts.

Students may also fail to recognize the connection between a designer's implicit biases and the way that these biases manifest in exclusionary designs. Designers, being human, inherently embed their biases and values into the artifacts they create [101], as well as their assumptions about users' capabilities, resources, and interaction styles [45, 70]. However, students do not always see the connection between the features of a design and the (implicit or explicit) beliefs of its designer. One reason for this is because software and other technical artifacts often carry perceptions of objectivity [18], making it difficult for early designers to understand the ways in which subjective choices manifest in design features. Another difficulty arises in moving students from considering only the features of a design to considering the broader contexts and systems of power that impact design decisions, especially if these contexts are never explicitly discussed or interrogated [268]. Either way, this challenge can lead to students misconstruing design bias as only a feature of an artifact itself, rather than a product of how the artifact's designer conceived of users or of the world during its creation.

Here, it is worth distinguishing the harmful design biases discussed in this chapter from the standpoints and subjectivities of designers themselves. No early designer approaches a design problem as a completely blank slate. They are informed by their prior understandings of the world and their lived experiences. In the realm of engineering design eduction, Svihla et al. recently explored how enabling students to draw up on their personal *funds of knowledge* might help first-year students engage more deeply with design problem framing [278]. First-year students who engaged with design problems that allowed them to draw upon their community, family, or recreation-based prior knowledge succeeding in framing problems in expert-like ways, even more so than senior design students who were given more traditional design problems. Funds of knowledge-based approaches like these allow students to integrate their expertise in areas other than design into design learning—a particularly notable benefit when considering the broader power structures at play within formal education and how traditional academic contexts can de-legitimize knowledge of people from various, often intersectionally minoritized groups [304, 305]. Designers' subjective perspectives are not the problem when it comes to teaching and learning inclusive design: Instead, issues of inclusion arise when students do not *recognize* the ways in which their perspectives and experiences differ from others', and thus make inaccurate assumptions about how their users move through the world, which can result in biased designs. The technique presented in this chapter aims to help students learn how designers' conceptions of their users concretely impact the inclusiveness of existing designs, so that students might later be more aware of their own biases during design processes.

A final challenge around teaching and learning inclusive design is that students can struggle to understand the perspectives of users different than themselves without resorting to stereotyping. Attempts to address this challenge often take the form of strategies, design methods, or tools that help students perspective-take or empathize with different kinds of users, such as through participatory/co-design methods [19, 198, 303, 315], use and co-creation of personas that focus on different aspects of users' identities [7, 24, 217], or variations of standard design evaluation methods such as cognitive walkthroughs [123, 208, 228]. Developing students' empathy skills is often seen as an important goal among HCI and UX design educators [302], though educators may find it difficult to support empathy development "amongst [the] young cognitively and physically high-performing students" that tend to dominate many computing and information science departments [192]. Early designers might also believe that they "just know" what different kinds of users want without having to do user research [268]. Students often may lack the prior experience or knowledge base needed to be aware of the ways that users with different capabilities, identities, and contexts might interact with designs, especially if they are part of one or more privileged groups [229, 244]. In the worst cases, this line of thinking can lead to stereotyping of marginalized groups during the design process: empathy-based methods may privilege the perspectives of the designer over those of actual users, leading to designs that embed biases and harmful stereotypes [19, 20, 37]. A successful method for teaching inclusive design skills should help early designers navigate these tensions, supporting productive empathizing while minimizing stereotyping behavior.

5.2.2 Design Rationale and the Role of Assumptions

Designers make decisions during their design processes that involve consideration of multiple variables, such as their understandings of the design space, its constraints, any given requirements, and user needs and preferences, among others. The reasoning behind these decisions is known as *design rationale*, and serves as justification for a designer's choices. Prior work from the area of software design suggests that documenting rationale during the design process can lead to higher quality final designs [41, 187] because it allows for designers to better account for artificial limitations they unintentionally placed on the design space [262]. Making design rationale explicit can also provide guidance for future design re-use efforts and concentrate organizational knowledge that might otherwise be diffuse or implicit in single, more easily accessible locations [184].

Because a single designer (or even a team of designers) can never have perfect information about the world, their users, or unanticipated interactions of either of these with their proposed design, they tend to rely on assumptions to make design decisions [189, 294]. Assumptions made during the design process are not always explicitly documented or even consciously recognized on the part of designers [34, 240]. As a result, these assumptions can be vectors for design bias, especially when designers assume certain things about potential users' ability levels, social or cultural contexts, or access to resources. Unless particular traits or characteristics about users are specified, technology designers tend to fall back on designing for users of socially dominant or majority races, genders, ages, cultures, and/or classes—even if the designer themself is from a historically marginalized group [70]. Explicitly surfacing and documenting assumptions made during the design process provides one way to catch potentially harmful biases and, ideally, to minimize their impacts on design decisions. This notion is supported by prior work on the role of assumptions within software rationale documentation, which suggests that augmenting rationale with explicit assumptions can help identify intervention points for improving a design, because it enables detection of parts of a system that rest on incorrect assumptions [41].

If assumptions are not caught during the early stages of the design process, they might also be identified and addressed using various design evaluation methods. Designers might turn to empirical methods such as usability studies, technology probes [149], or experience sampling techniques [65] to better understand how potential users might interact with their designs in more authentic contexts. The information gained through these methods can reveal hidden assumptions that were made during the design process which break down upon exposure to different interaction styles or preferences. Empirical design evaluation techniques can be used to test the efficacy or utility of a design, but they generally frame these findings around the designer's conceptions of their users. As a result, though empirical evaluation methods can reveal information that signals the existence of inaccurate user assumptions in a design, they do not support students in explicitly identifying the assumptions themselves or in drawing connections between embedded assumptions, design bias, and design decisions.

Alternately, designers might also employ analytical design evaluation methods to surface assumptions that are embedded within their designs. For instance, the claims analysis aspect of scenario-based design [48, 49] provides a causal mechanism for tying design features to design rationale. Claims analysis can help designers articulate ways in which different aspects of their design afford (or preclude) various outcomes and reactions on the part of users, and might serve to reveal embedded assumptions about users' environments, contexts, or preferences, though the core method does not explicitly reference inclusion as a design goal. Analytical methods like heuristic evaluations [220] and cognitive walkthroughs [298] might also help reveal erroneous assumptions designers made about a user's prior knowledge or preferred interaction styles.

Another analytical method designers might use to understand their users is the empathy map [155]. Several variations of empathy mapping techniques exist, but the key goal of empathy maps is to make users' underlying traits apparent by cataloguing what a user might see, think, feel, or otherwise experience [267]. Empathy maps may help reveal gaps in design rationale, especially if they are informed by primary user research or co-developed with users, and can certainly illustrate alternate perspectives that designers may not have considered previously, which may positively impact inclusion. However, the empathy map method does not explicitly foreground the role of assumptions in the design process, which may not support early designers in making critical connections between designers' conceptions of users and their final design's features.

When targeting inclusive design in particular, designers might use modified analytical evaluation methods which specifically aim to evaluate gender [277], socioeconomic status [208], or cultural [7] inclusiveness. There is a small yet growing body of work that investigates how educators might use single-facet inclusiveness-focused analytical evaluation methods in the classroom, such as Oleson et al.'s GenderMag-Teach effort [228], or Anvari et al.'s investigation of cross-cultural persona use in a large user-centered design class [7]. However, in general, the analytical design evaluation methods described above were not originally designed for education purposes. Existing inclusiveness-focused analytical evaluation methods are not well-suited to address the particular problems students face when learning inclusive design skills (described in the previous section), nor are they targeted at early designers who are just learning the basics of the discipline. Further, similar to empirical design evaluation methods, these methods rarely frame their insights in terms of assumptions about users or provide the scaffolding learners need to tie assumptions, design bias, and inclusion together. In contrast, my proposed technique was created to be used in educational contexts, directly addressing several challenges and misconceptions students face when learning identify assumptions and design inclusively.

5.3 The CIDER Assumption Elicitation Technique

To help students learn to design more inclusively, I created the CIDER assumption elicitation technique (Figure 5.2), which stands for *Critique, Imagine, Design, Expand, Repeat*. This technique leverages guided critique, brainstorming, and feedback to help students understand how biased assumptions can manifest in design features and exclude people from interacting with a design as

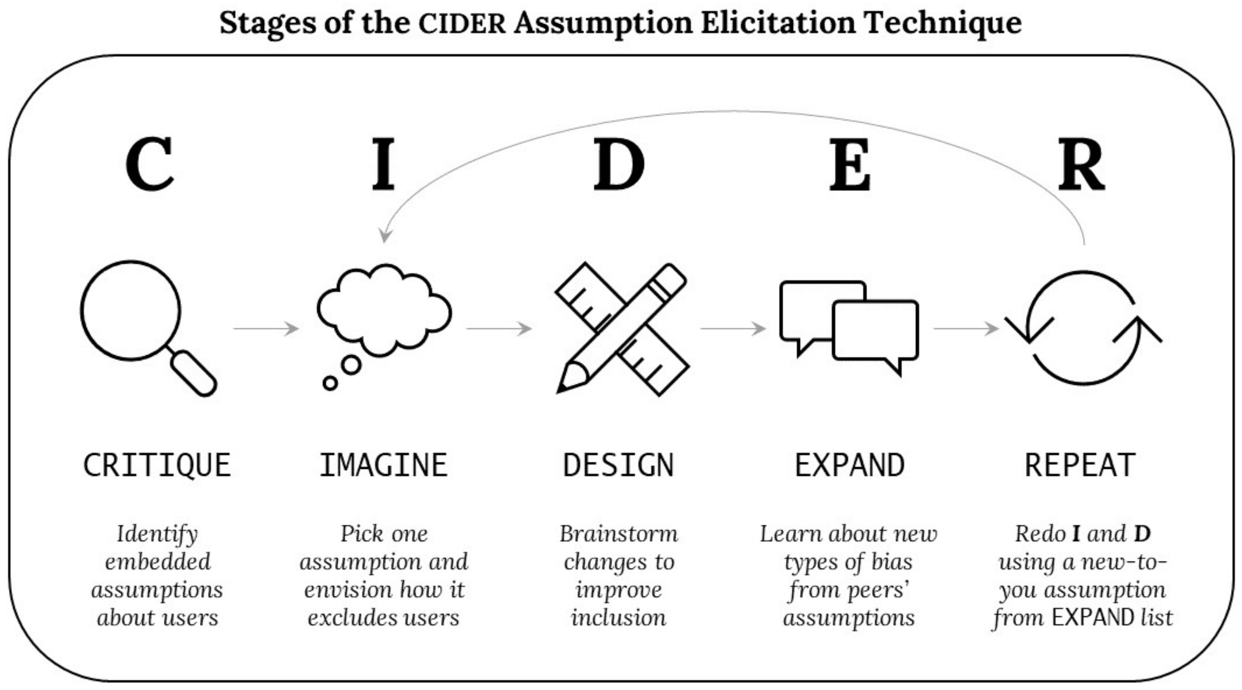

Stages of the CIDER Assumption Elicitation Technique
C
I
D
E
R
CRITIQUE
Identify embedded assumptions about users
IMAGINE
Pick one assumption and envision how it excludes users
DESIGN
Brainstorm changes to improve inclusion
EXPAND
Learn about new types of bias from peers' assumptions
REPEAT
Redo **I** and **D** using a new-to-you assumption from EXPAND list

Figure 5.2: The five stages of the CIDER assumption elicitation technique for helping novice designers learn to recognize and respond to design bias. For each CIDER-based activity, the educator chooses an artifact for the whole class to analyze. Then, students use the five CIDER stages to identify assumptions about users present within the design, understand how those assumptions might lead to exclusion, practice brainstorming inclusive redesigns, and broaden their knowledge bases of design bias by engaging with peers' CIDER responses.

intended.

The goal of CIDER is to help students identify the ways that designers' implicit or explicit assumptions about user ability, capacity, environment, or resources concretely manifest in and contribute to exclusionary interface designs. The technique I describe here was originally intended for use in post-secondary design contexts that emphasize technology design and development, such as a HCI or UX design class, but the underlying principles may transfer to other contexts like K-12 computing education as well. A complete CIDER-based activity consists of the educator choosing an artifact of analysis, and then five major stages students progress through (see Figure 5.2). If time constraints require, educator might only conduct a single CIDER activity in their course—according to my case study, even a single CIDER activity may still contribute to better understandings of inclusive design. However, I found the best results to occur over time with reinforcement and repetition, conducting a series of CIDER activities over a span of several weeks and using a different artifact of analysis each time. This multiple-use approach enabled consideration of a broader range of assumptions, because different artifacts' interaction styles can make different types of assumptions more or less salient to students[4].

As described in Section 5.2.1, students may face a number of difficulties when learning and practicing inclusive design skills. The CIDER technique was designed to target five of these challenges in particular:

1. **Motivating inclusion:** Students may devalue design work or believe inclusion issues do not really exist in "real-world" designs, lacking motivation to learn and practice inclusive design [55, 130, 193, 228, 229].

2. **Connecting features to assumptions:** Students, especially those with little design experience, may not recognize the connection between exclusionary design features and a designer's assumptions about users' capabilities and contexts [18, 66, 70, 101, 268, 307, 309].

3. **Designing for diversity:** Students may implicitly design for users as a homogeneous population, failing to recognize and account for diversity, especially if they do not have extensive knowledge bases of user experiences to draw upon [7, 70, 192, 219, 228, 268, 277].

4. **Acting on inclusion goals:** Even if they value it already, students may struggle to move from abstract appreciations of inclusion as a goal to concrete design actions they can take to reduce or mitigate design biases [54, 125, 228, 229, 295].

[4]For instance, I found that my participants only surfaced assumptions related to users' potential hearing ability when the artifact under analysis had a strong audio-based component to its interactions (Zoom video calling software, Google Home voice assistant). However, students identified assumptions related to users' potential visual ability across all activities.

5. **Avoiding stereotyping:** Students may struggle to understand the perspectives and experiences of users who are unlike themselves without resorting to stereotyping [20, 192, 229, 244, 268, 302].

Below, I describe how the CIDER technique addresses each of these challenges, beginning with the educator's choice of artifact and progressing through the five stages students engage in as they progress through the activity (Figure 5.2). For clarity, I provide a running example throughout the following section, using a common QWERTY desktop keyboard as a basis and providing illustrative quotes from participants in my case study. For the purposes of this chapter, when referring to the five named stages of CIDER, the stage names appear capitalized and in monospace font (e.g. `CRITIQUE`).

5.3.1 Set-up: Educator selects a real-world HCI artifact for students to analyze

In preparation for conducting one of these activities, the educator first chooses some existing, real-world HCI artifact for students to critique using the CIDER technique process. This artifact could be any piece of software or hardware that has an interface and/or affords user interactions, but ideally should be a piece of technology that students are aware of, and possibly that they have interacted with before. For instance, some of the artifacts I used for CIDER activities in my case study included a Google Home digital voice assistant, the Zoom video calling desktop software interface, and an informational webpage from the university's website.

Once the educator chooses an artifact, they then find a way of conveying the design's features and interactivity to students in a way that fits the medium of instruction. For smaller, in-person courses, the educator might bring in an example of the artifact for students to interact with before the activity. In larger or remote learning courses, the educator might gather some images of the artifact or direct students to a electronic prototype. For the purposes of the case study, which was conducted during a period of exclusively remote learning necessitated by the COVID-19 pandemic, I found promising results using a combination of images of the artifact (one or more screenshots or pictures showcasing different functionality) along with a basic textual description of the device or interface's most salient features. For instance, the description of the desktop keyboard used as an example throughout this section highlighted the standard QWERTY layout of the board's English characters and symbols, its ten key number pad, the indicator lights in its upper right corner, and the board's USB wired connection which allows people to input text and command sequences to connected devices.

Challenge addressed: Motivating inclusion (1)

Critiquing a real-world artifact and uncovering biases and assumptions present within its design helps students begin to understand that no design is infallible, and that inclusion issues truly do exist in the products around them. This can serve to motivate resistant learners [228] who may be skeptical of the existence or severity of design inclusion issues.

This also provides an opportunity for students to connect their own prior experiences and expertise to course content, which can increase engagement and motivation for learning inclusive design concepts [29, 215]. Using existing artifacts rather than students' own designs as the object of analysis also means that CIDER activities can be assigned and completed earlier in the course, conceivably as early as the first or second session of instruction, as there is no need to wait for students to gain enough design competence to create something of their own. When integrated into a course's early stages, the explicit focus on inclusion present in CIDER activities can even help to set a tone of considering human diversity when designing, contributing to foundations of inclusive design knowledge. Finally, because early designers may struggle to objectively and accurately critique their own designs, focusing CIDER activities on artifacts designed by others first can help circumvent blind spots obstructing recognition of biases.

5.3.2 C: Students `CRITIQUE` the artifact's design to identify embedded assumptions about users

Once the artifact of analysis for a CIDER activity is chosen, students can begin to identify its embedded assumptions. In the `CRITIQUE` stage, students list as many assumptions about users' potential capabilities, contexts, environments, and/or available resources as they can identify within

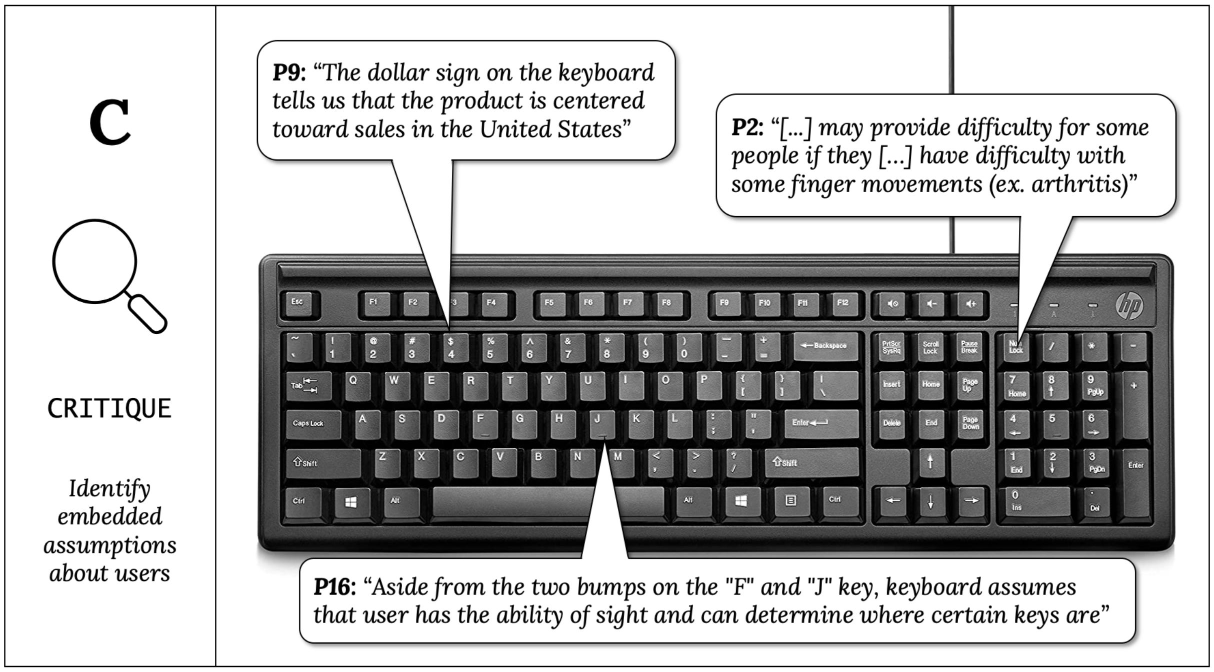

C
CRITIQUE
Identify embedded assumptions about users
P9: "The dollar sign on the keyboard tells us that the product is centered toward sales in the United States"
P2: "[...] may provide difficulty for some people if they [...] have difficulty with some finger movements (ex. arthritis)"
P16: "Aside from the two bumps on the "F" and "J" key, keyboard assumes that user has the ability of sight and can determine where certain keys are"

Figure 5.3: Stage 1 of CIDER: `CRITIQUE`. Students draw on their prior knowledge and experiences to identify ways in which a design might rest on assumptions about users.

the design. In my case study, for this stage, I used the prompt *"What assumptions do the designs make about users' potential interactions with the devices? List as many as you can think of in the next 3-5 minutes or so (bullet points encouraged)"*, along with an example assumption provided by the educator for clarity. Illustrative examples of students' responses can be found in Figure 5.3. In this stage, students should be encouraged to identify many different types of assumptions rather than focusing on in-depth descriptions of one or two assumptions. This helps resist fixation and encourages students to draw deeply upon their own experiences, observations of others, or other prior knowledge to identify how a design's features might present barriers to users.

Challenge addressed: Connecting features to assumptions (2)

Explicitly framing the shortfalls of a design in language like *assumptions* made by designers about users, *bias*, *inclusion*, and *exclusion* is key to addressing student learning challenges around how assumptions might manifest in design features. Using this frame helps to dispel notions of objectivity about design that early designers may hold [229], such as the belief that there is a "correct" design that will work for everyone. This emphasis on subjectivity highlights how designers' conceptions of their users influence their design decisions, making the connection between a designer's assumptions, design bias, and exclusion apparent. Using assumptions as a lens for critique helps students keep in mind that HCI artifacts are used in real-world contexts by a wide array of people, not by stereotypical "average" users or in "ideal" conditions. It also exemplifies the responsibility students might have as future designers to consider a wide range of user capabilities and contexts: If they are aware of the impacts their design decisions might have, they may be more inspired to value inclusion and interrogate their own assumptions within their own design processes.

5.3.3 I: Students `IMAGINE` how a particular assumption might lead to exclusion

Once students have identified some assumptions about users within a design, they can begin to consider how the design might be inaccessible to some users. In the `IMAGINE` stage, students choose an assumption from the list they created in the `CRITIQUE` stage. Then, they come up with a short scenario in which the artifact's design breaks down, describing how a user for which the chosen assumption is inaccurate might not be able to interact with the artifact as its designers intended. My case study's CIDER activities used the prompts *"Select one of the above assumptions that you think is important to address"* and *"Write a 1-2 sentence scenario where a user could not use the [artifact] as expected because of the assumption you selected. This represents one way the design could exclude certain users."* One participant's chosen assumption and imagined scenario of exclusion for a QWERTY desktop keyboard can be found in Figure 5.4.

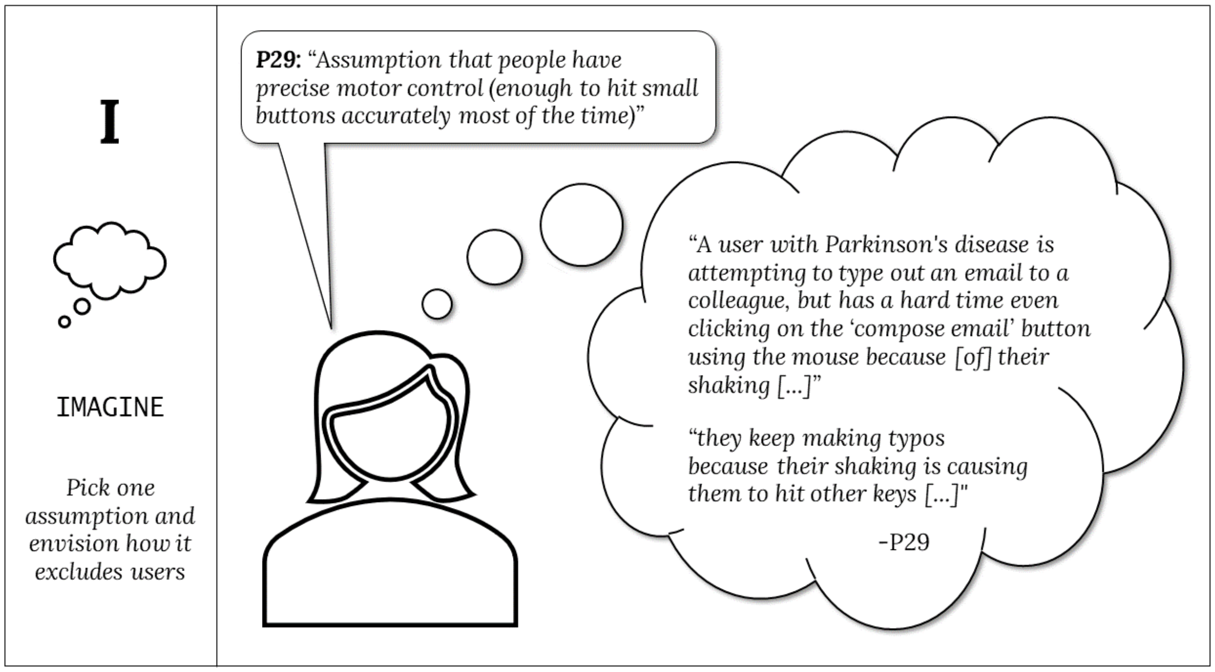

I
IMAGINE
Pick one assumption and envision how it excludes users
P29: "Assumption that people have precise motor control (enough to hit small buttons accurately most of the time)"
"A user with Parkinson's disease is attempting to type out an email to a colleague, but has a hard time even clicking on the 'compose email' button using the mouse because [of] their shaking [...]"
"they keep making typos because their shaking is causing them to hit other keys [...]"
-P29

Figure 5.4: Stage 2 of CIDER: IMAGINE. Students choose one assumption they identify and describe a scenario in which that assumption prevents a user from engaging with or using the design.

Challenges addressed: Designing for diversity (3), Avoiding stereotyping (5)

Asking students to imagine a scenario where the design breaks down for a particular user helps refute the implicit misconception some students may have that users are a monolithic, homogeneous population, addressing challenges students might have around how to design for diversity. The scenario students imagine is a direct counterexample to the inaccurate notion that what works well for one user can or should work well for everyone. Further, framing design bias and exclusion as scenarios of usage centers the actual people affected by the artifact's design. As mentioned previously, it can also be difficult for students to properly empathize with users unlike themselves without stereotyping. To circumvent this, the IMAGINE stage asks students to focus on the concrete impacts that the design bias might have on a person that could prevent them from fully

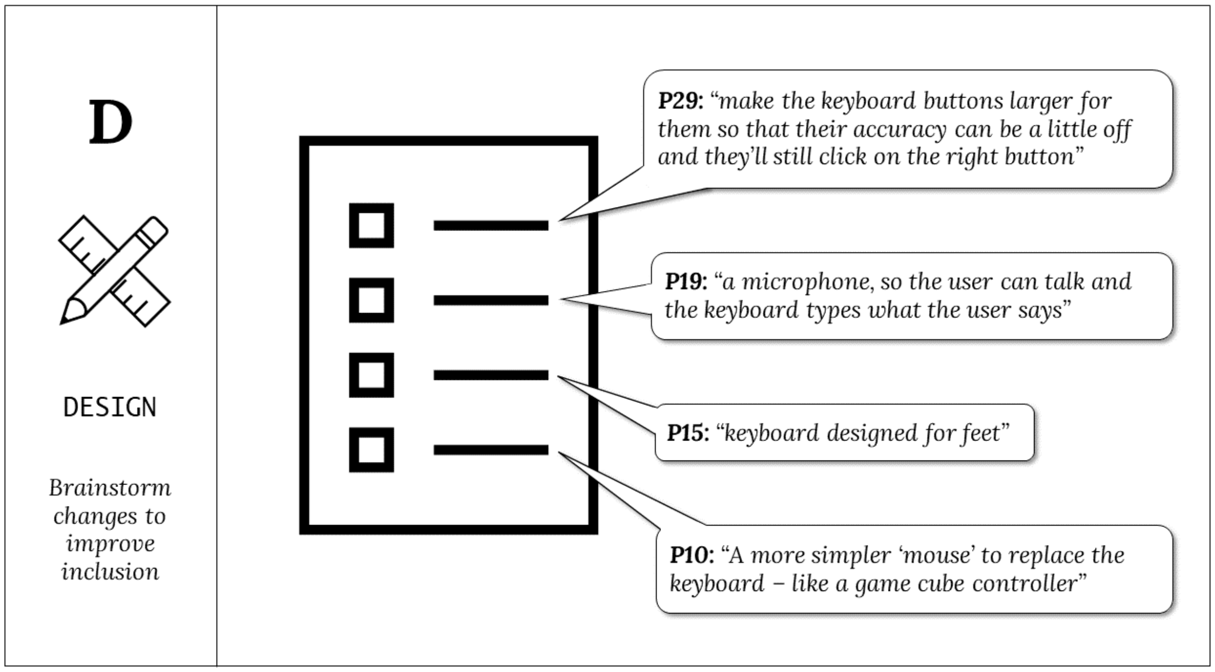

D
DESIGN
Brainstorm changes to improve inclusion
P29: "make the keyboard buttons larger for them so that their accuracy can be a little off and they'll still click on the right button"
P19: "a microphone, so the user can talk and the keyboard types what the user says"
P15: "keyboard designed for feet"
P10: "A more simpler 'mouse' to replace the keyboard – like a game cube controller"

Figure 5.5: Stage 3 of CIDER: `DESIGN`. Students brainstorm several changes to the design that would address the scenario they came up with in the `IMAGINE` stage, removing barriers to access. Students are encouraged to think of as many ways to improve the design's inclusion as they can, regardless of potential feasibility.

interacting with a design as intended. In this way, this stage of the technique provides some scaffolding for understanding the ways in which someone might not be able to interact with a design due to bias, while resisting students' potential unintentional reductions of minoritized populations to stereotypes.

5.3.4 D: Students practice `DESIGN` by brainstorming ways to address the assumption and make the artifact's design more inclusive

After envisioning a scenario of exclusion in the `IMAGINE` stage, students have a concrete starting point from which to begin thinking about improving a design's inclusion. In the `DESIGN` stage of the CIDER technique, students brainstorm ways to change or adapt the artifact's design which would circumvent the scenario they described, listing as many as they can. My case study's CIDER activities used the prompt *"Brainstorm ways to change the design of the [artifact] to avoid the scenario you wrote above. List as many different kinds of potential solutions you can think of over the next 3-5 minutes – aim for ten or more. Bullet points encouraged."* Several participants' responses to how one might modify or change a QWERTY desktop keyboard to avoid relying on the assumption that a user has fine motor control in their hands can be seen in Figure 5.5.

Challenge addressed: Acting on inclusion goals (4)

By brainstorming with assumptions about users in mind, students practice coming up with actionable ways to address design bias and improve inclusion, helping them transition from abstract inclusion goals to concrete design actions. Targeting a specific assumption and a specific scenario of exclusion helps reduce uncertainty about what inclusion means or how to increase inclusiveness of a design. Further, asking students to come up with multiple different ways to address one assumption underscores that there is not necessarily a single "correct" answer to improve the artifact's inclusiveness. For simplicity's sake, the CIDER activity generally frames inclusion as reduced or mitigated barriers to use which in turn increases access for more user populations, similar to definitions used by Keates et al. [167, 168] and Goodman-Deane et al. [121], However, a design modification which increases access for one user does not necessarily lead to increased access for all users. Asking students to brainstorm several solutions encourages consideration of tradeoffs and constraints which might make some solutions more well-suited to addressing an inclusiveness issue than others.

5.3.5 E: Students `EXPAND` *their understandings of inclusive design by engaging with peers' responses*

While progressing through the previous stages of CIDER, students produce two lists—one of identified assumptions, one of brainstormed changes—and a scenario of potential design exclusion. In

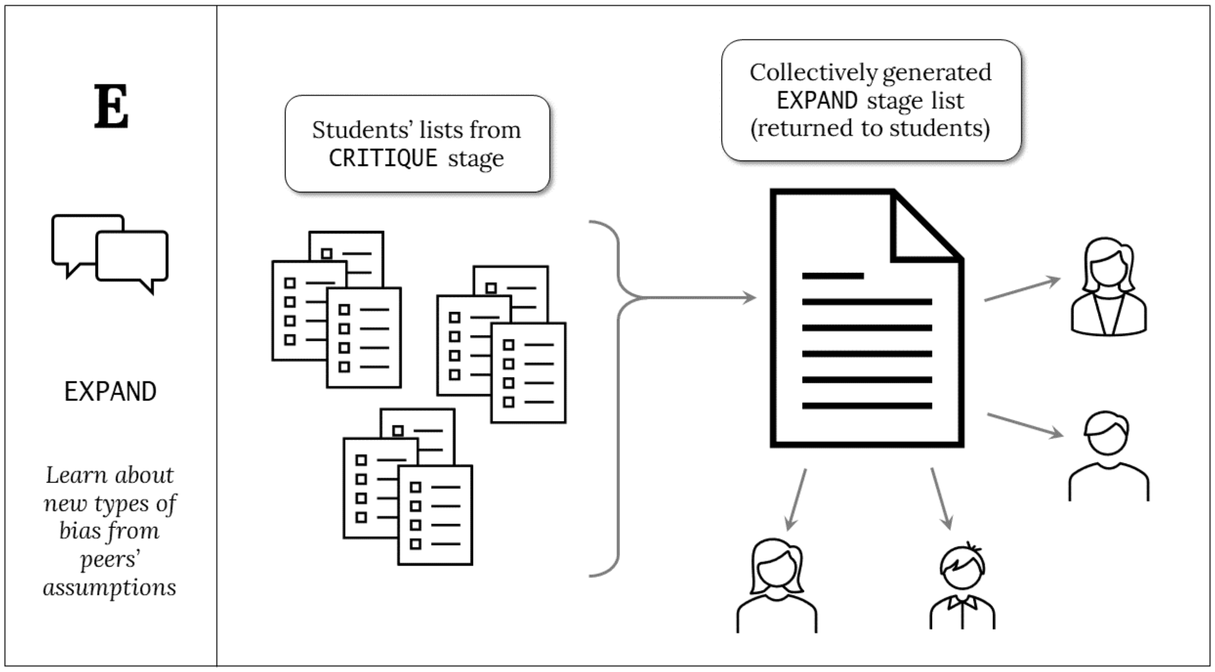

E
EXPAND
Learn about new types of bias from peers' assumptions
Students' lists from CRITIQUE stage
Collectively generated EXPAND stage list (returned to students)

Figure 5.6: Stage 4 of CIDER: EXPAND. The instructor collects the lists of assumptions students came up with during the CRITIQUE stage and uses them to create an overall list of assumptions embedded into the design, augmenting with their own expertise when necessary. The EXPAND list is made accessible to students and integrated into the next stage's activity.

the EXPAND stage (Figure 5.6), the educator collects and combines students' lists of assumptions from the CRITIQUE stage to create a collective, more complete list of the assumptions embedded within the artifact's design. The goal of the EXPAND list is to be a single, shared resource that details a wide breadth of assumption types, giving an overview of the *assumption space* for the artifact. In this way, the collectively-generated EXPAND list serves as a way for students to learn about new types of bias from their peers' responses and consider assumptions that they had not identified in their own work. Representative assumptions can be copy-pasted directly from students' submissions to build the list, or they can be paraphrased for clarity as needed. Once this list is made, the educator shares it back to the class and integrates it into the final part of the

CIDER activity. In my case study, the educator accomplished this by collecting students' lists of assumptions, then manually reviewing and combining them into a single list, removing duplicates and augmenting the list's assumptions with their own expertise if there were any obvious gaps. The educator then posted the `EXPAND` list to the class's shared cloud storage space, encouraging students to review the list once it was available, and integrating it into the final part of the activity.

Challenge addressed: Designing for diversity (3)

This stage serves as a key source of feedback for students, which is important for promoting learning [29]. The `EXPAND` list provides a means for students to compare the assumptions they identified on their own against the more complete list comprised of educator and peer-identified assumptions, revealing new perspectives they had missed and expanding their awareness of the assumption space for the artifact. Engaging with the breadth of assumptions covered in the `EXPAND` list helps students to build their knowledge bases of user experiences and design exclusion. By reflecting on the perspectives of their peers, students become aware of more ways users' capabilities and contexts can differ and how they might then design for different experiences.

5.3.6 R: Students `REPEAT` stages `IMAGINE` and `DESIGN` using a peer's assumption

To complete the CIDER technique, students draw upon their peers' knowledge bases and consider how a new type of design bias might lead to exclusion. In the `REPEAT` stage, students review the collectively-generated list of assumptions from the `EXPAND` stage and select one that they feel is important to address, but that they had not surfaced themselves during their own previous critique of the artifact. They then repeat the steps of the `IMAGINE` and `DESIGN` stages using this new assumption as a focus. My case study's activity included the prompt *"Select another assumption from the list above that you think is important to address. Make sure to choose a different assumption than you used for [previous critique]. Choose one that you didn't even come up with during [previous critique], if possible"* for this stage, followed by similar prompts to those already described in the `IMAGINE` and `DESIGN` subsections. An example of a participant's chosen

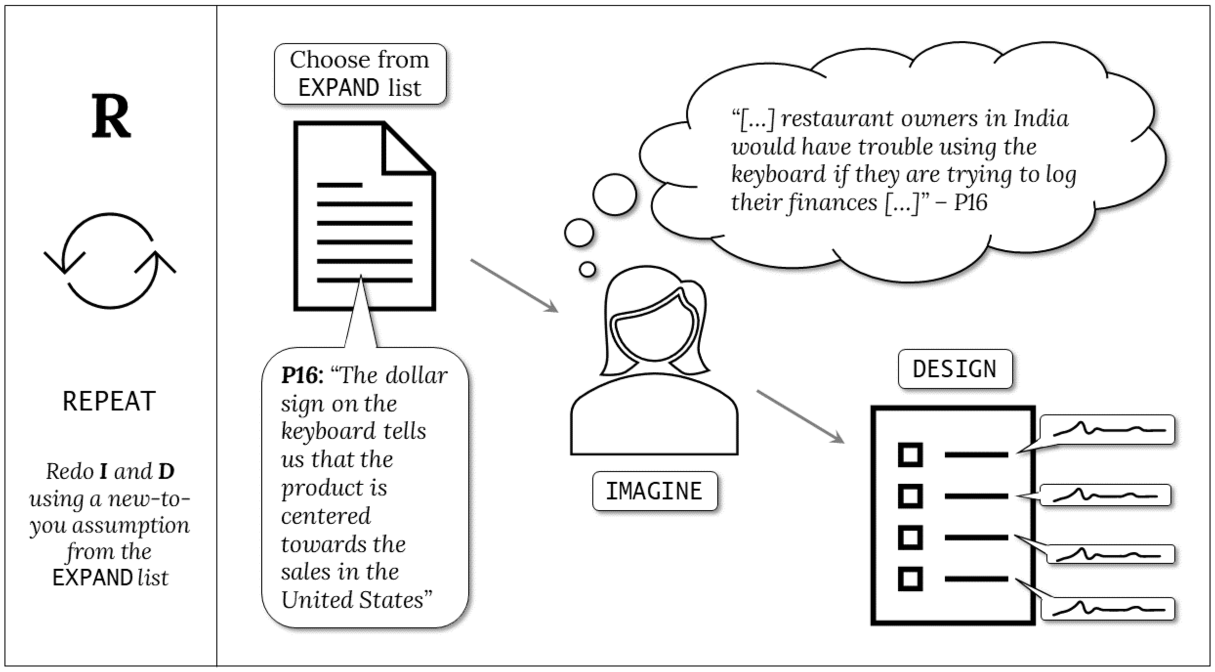

R
REPEAT
Redo I and D using a new-to-you assumption from the EXPAND list
Choose from EXPAND list
P16: "The dollar sign on the keyboard tells us that the product is centered towards the sales in the United States"
"[...] restaurant owners in India would have trouble using the keyboard if they are trying to log their finances [...]" – P16
IMAGINE
DESIGN

Figure 5.7: Stage 5 of CIDER: `REPEAT`. Students review the `EXPAND` list and select an assumption which they had not identified during their own `CRITIQUE` activity. Students then go through the `IMAGINE` and `DESIGN` stages again using the new assumption.

assumption from the `EXPAND` list and their accompanying scenario of exclusion can be found in Figure 5.7.

Challenges addressed: Connecting features to assumptions (2), Designing for diversity (3), Acting on inclusion goals (4), Avoiding stereotyping (5)

Repeating the middle stages of the CIDER technique with a new assumption gives students more practice imagining scenarios of design and brainstorming concrete ways to address design bias. By using an assumption they had not identified themselves previously, students are also guaranteed to expand their knowledge of how design exclusion manifests by at least a single instance of bias. The repetition of these stages helps reinforce the connections between designers' assumptions and potential design bias, diversifying students' knowledge bases concepts and helping them generalize to broader understandings of inclusive design principles.

5.4 Case Study: Method

To evaluate the efficacy of the CIDER assumption elicitation technique, I conducted a case study in an introductory design methods course which spanned eleven weeks of instruction and concluded with follow-up interviews a month after the course's conclusion. The technique was integrated into the course through five individual CIDER activities using different artifacts of analysis, and one team CIDER activity. I followed the concurrent embedded approach to mixed method research [72], collecting quantitative data to explore students' changes in design self-efficacy (RQ1) alongside qualitative data to understand students' experiences with both individual (RQ2) and collaborative (RQ3) CIDER activities, and supplementing these understandings with qualitative analyses of post-course interviews (RQ4). The study explored the following research questions:

1. How might CIDER-based activities impact students' self-efficacy as a designer?
2. How might the CIDER technique help students recognize different types of exclusionary design biases?
3. How might conducting CIDER-based activities collaboratively, rather than individually, impact students' experiences?
4. What kinds of lasting impacts might the CIDER technique have on students' design approaches?

5.4.1 Study Context

Course context: Accessibility-heavy design culture; Remote learning

The course I used for my case study was an undergraduate introductory design methods course in the information science department at the University of Washington, a large, public, United States-

Week	Topics Covered	CIDER Activities
1	Class structure; What designers do	
2	Design process; Understanding problems (First mention of accessibility/inclusion)	QWERTY keyboard and mouse
3	Defining problems; Brainstorming	Zoom video calling software
4	Sketching; Prototyping; Interface design	University COVID-19 info site
5	Critique; Empirical evaluation	Google Home voice assistant
6	Analytical evaluation; Midterm	Revo R180 touchscreen toaster
7	Project: Research, Problem definition	
8	Project: Peer critique, Brainstorming, Low-fi prototyping	
9	Project: Peer critique, Feedback from users, Iteration	
10	Project: Evaluation, Limitations, High-fi prototyping	Collaborative activity on project prototype
11	Project: Design specification submission	

Table 5.1: The schedule of the course, including topics covered each week. Students did five individual CIDER activities on different artifacts from Week 2 to Week 6 of the course, then a collaborative CIDER activity on their own prototypes in Week 10. Due to the nature of instruction, the concepts of accessibility and inclusion were introduced early on in the course and integrated throughout many of the topics.

based university. The course focused on the design of user interfaces and interactive hardware and software-based systems. The instructor of record for this course was the dissertation author, who had one prior term of teaching experience with the course. Based on their own background, the instructor taught from the perspective that designs are never value-neutral, and as a result that designers have a responsibility to carefully consider the interaction styles and preferences of many different types of users throughout their design processes. Given this, the notion of accessibility and inclusion was introduced early on through course readings and integrated throughout many of the topics and design exercises that followed.

The particular university at which the study took place has a strong design culture in its computer and information science departments and tends to emphasize human-computer interaction

(HCI) and accessibility in its research focus. While this creates a favorable environment to deploy and explore inclusive design learning techniques, it also means that the students at this university may be more aware of or open to inclusion-related topics than students elsewhere. It is possible that the case study would have produced different results if conducted at an institution whose technical departments were less favorable toward design and HCI. Future work should explore this possibility, evaluating the utility of the CIDER technique across a broad variety of learning contexts.

The course was taught during one of the first fully remote teaching terms necessitated by the COVID-19 pandemic. This required several changes to the typical course structure to ensure that students could engage with all the course elements entirely online. For instance, due to inherent inequities in requiring synchronous remote learning, the instructor allowed for students to participate in discussions either in synchronous small groups over video calling platforms like Zoom, or on asynchronous class discussion boards. It also required us to adapt the CIDER activities to a format that worked for remote learning. As I describe later on, I did still see strong evidence of inclusive design learning even in remote learning contexts, though future work should investigate potential benefits and tradeoffs of conducting CIDER activities in-person or online. The above factors may be considered limitations to the interpretations of my findings, but also additional context to better understand the backdrops against which my results arose.

Course structure

There were no prerequisites for the course, so I assumed no particular level of prior design knowledge or experience on students' parts. However, in practice, most students had taken a prior introductory course within the major that had students apply the basic steps of the technology design process for a course project. The course I conducted this study in covered topics from the basics of what design entails and what roles designers play in making a product, to how to work through the various stages of the design process (using the Design Thinking framework [46] as foundation for instruction). A detailed schedule of topics can be found in Table 5.1 alongside a timeline of the course, which spanned eleven weeks (ten weeks of instruction plus one week of final exams).

The first part of the course (Weeks 1-6) followed a flipped classroom paradigm in which students read material about design foundations on their own and then participated in class discussions about the topics. As mentioned previously, many of these discussions took place on electronic peer discussion boards due to the nature of remote learning. To gain experience with design work, students also did *deliverables*, weekly activities where they practiced design techniques (ideating, brainstorming, critiquing, etc.) synchronously with a partner.

To give students experience designing alongside others, introductory design courses often include team projects as an aspect of instruction. My course integrated a final project during the latter part of the class (Weeks 7-11). In teams of 2-3 members, students were tasked with coming up with an original design concept, moving through design stages from user research, to brainstorming and ideation, low and high-fidelity prototyping, and iterative critique over the course of four weeks (see Table 5.1 for details of timing). The theme of the projects for the course was to address an information gap that contributed to systemic inequality on campus or in the surrounding city. Students practiced communicating their design processes in the form of a design specification, where they wrote up the results of their user research, described their design concept and their design evaluation processes, and finally elaborated on any known limitations of their design concept.

Participants

There were 40 students enrolled in the course. Students could enroll at any undergraduate class standing and without necessarily needing to be in the information science major, though the course was required for students enrolled in the major. Thirty-two students self-reported computer or information science as their current or intended major field of study. Four students reported that computer or information science was their current or intended minor (with majors in other disciplines). Three students self-reported other major fields of study, and one student declined to report this information.

I also asked to students to describe any prior design experience they had. Nineteen students self-reported that the only design experience they had was from prior classes (one of the courses traditionally taken before this one was a survey course which included a small design project).

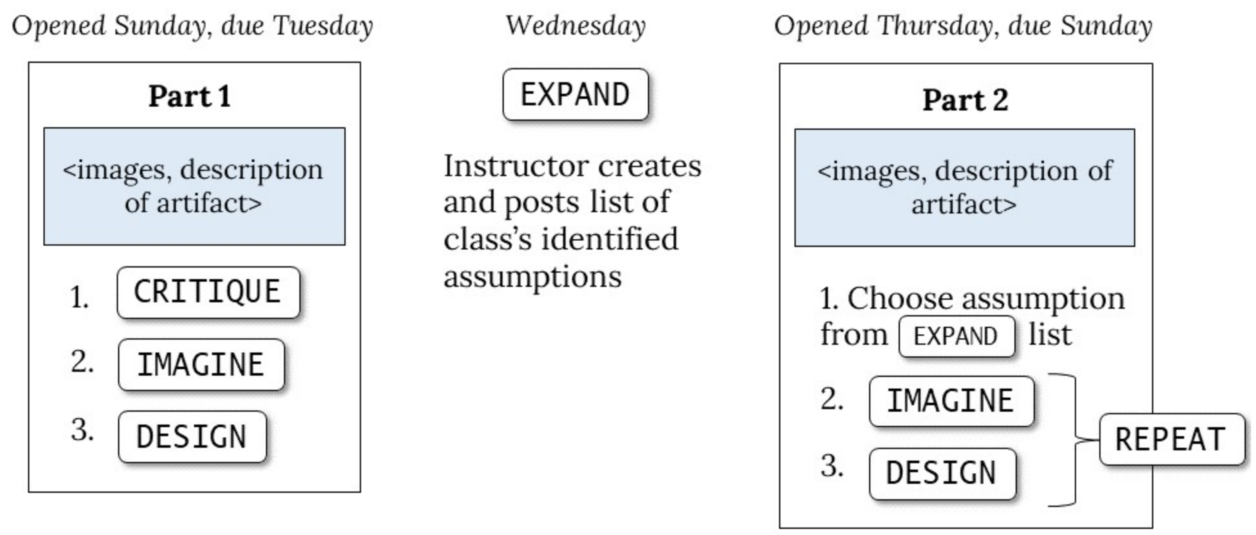

Opened Sunday, due Tuesday
Part 1
<images, description of artifact>
1. CRITIQUE
2. IMAGINE
3. DESIGN
Wednesday
EXPAND
Instructor creates and posts list of class's identified assumptions
Opened Thursday, due Sunday
Part 2
<images, description of artifact>
1. Choose assumption from EXPAND list
2. IMAGINE
3. DESIGN
REPEAT

Figure 5.8: The structure and timing of the individual CIDER activities used for the case study in Weeks 2-6 of instruction. The wording used for each stage's prompts can be found in the corresponding subsection of Section 3.

Ten students reported that they had done design for their personal, non-class projects, such as creating an interface for a website or mobile application. Six students reporting having some professional experience with design such as an internship. Only five students reported having no design experience prior to the course. To get a sense of students' perceived design expertise, I additionally asked them to fill in the blank of the statement *"I consider myself to be a <blank> designer"* with one of five options. The majority of students considered themselves *novice* (14) or *between novice and intermediate* (18) designers. 4 students considered themselves *intermediate* designers, and 2 considered themselves *between intermediate and expert*. No students considered themselves *expert* designers.

CIDER activity integration

Throughout the course, students completed six CIDER activities: five individual and one collaborative. Due to the remote nature of the course, all the activities were electronic-based formats hosted on the Canvas learning management system, the details of which are described below. To mitigate potential response bias on students' parts, all CIDER activities were graded only on completion—As long as students filled out each question of the activity and submitted it by the deadline, they automatically received full credit. The instructor disclosed to students that their responses to CIDER activities would be analyzed as part of a research study through the syllabus, through a addendum on the activity description itself, and during the first optional synchronous class meeting as they gave an overview of the course structure. In all these cases, students were made aware that they could elect for their responses not to be analyzed for research purposes by notifying either the instructor or the TA (who was unaffiliated with the research project) at any point before the end of the term, and that this choice would not have any impact on their grades or other personal course outcomes. No students opted out of participation in the study.

Individual CIDER activities. From Weeks 2 to 6 of instruction, students completed CIDER activities individually (see Table 5.1). Figure 5.8 shows the timing and generalized structure of these CIDER activities, with the tags (`CRITIQUE`, `EXPAND`, etc.) referencing the corresponding stages described in Section 5.3. Part 1 of each activity opened to students on Sunday afternoon, was due by the following Tuesday, and contained activity prompts for the `CRITIQUE`, `IMAGINE`, and `DESIGN` stages. The instructor compiled the week's `EXPAND` list of student-generated assumptions each Wednesday, posting the results in the class shared Google Drive and embedding the list into the Part 2 assignment for easy access. Part 2 of each CIDER activity opened on Thursday, was due the following Sunday morning, containing the prompts pertaining to the `EXPAND` list and the `REPEAT` stages of the technique.

Each CIDER activity used a different HCI artifact of analysis, chosen to represent a variety of interaction types, interface styles, and usage contexts.

- Week 2: A QWERTY desktop keyboard and mouse designed for use with a Windows computer, chosen as an example of physical hardware that required lots of fine motor interaction.
- Week 3: The Zoom video calling platform desktop interface, chosen as an example of a relatively resource-heavy software that students were familiar with and that required video

and audio-based interactions.

- Week 4: An informational webpage published by the university about COVID-19 and related classroom policies, chosen as an example of a software. interface with a very information-dense structure

- Week 5: The Google Home digital voice assistant device, chosen as an example of an artifact that relied largely on audio-based interactions rather than visual components.

- Week 6: The Revo R180, a toaster with a digital touchscreen interface, chosen as an example of a non-standard touch-based interface intended for use in a context where HCI artifacts are generally less common.

These individual activities were implemented as timed quizzes on the Canvas learning management platform. I chose to enforce a time limit of 30 minutes on each CIDER activity to avoid students spending excessive amounts of time on it during remote instruction, because these activities were originally intended to take no more than 10-15 minutes when integrated into a more traditional in-person classroom setting.

Collaborative CIDER activity. To explore the impact that collaboration and teamwork might have on students' experiences using the CIDER technique to identify potential design bias, I created a version of a CIDER activity that could be done in teams. The collaborative CIDER activity might be considered more similar to a professional design context, in which designer teams critique and evaluate their own artifacts, considering the needs of different groups of stakeholders. Project teams completed this collaborative activity during the final week of instruction (Week 10, see Table 5.1). The collaborative activity was positioned within the class as one way for teams to identify limitations of their designs, which they were required to report on in their final design specification writeups. To mitigate potential response bias from teams, the instructor took the same measures as they did for the individual CIDER activities. Additionally, the instructor made clear to teams that they would not view teams' collaborative CIDER activity responses and feedback until after final grades had been submitted.

The collaborative CIDER activity was similar in concept to the previously described individual CIDER activities, though it had several notable differences in format which enabled us to investigate the CIDER technique's utility in this new context:

- It was performed in teams, rather than individually. Teams consisted of 2-3 students, save for three students who opted to complete their projects individually. (Data from the students who worked individually was excluded from analysis for the purposes of this research question, because I was interested in collaborative aspects.)

- Instead of performing the activity on an existing artifact which was designed by someone else, teams were asked to use the CIDER technique on their own high-fidelity prototypes.

- As teams were identifying assumptions in their own designs and each team's artifact of analysis was different, the instructor did not create collective assumptions lists (i.e. the CIDER `EXPAND` stage was not present). However, all previous lists from the previous `EXPAND` stages of the individual CIDER activities were available to students for reference, if they wished to use them as a resource (and, as described later, many did).

- As a result of the above bullet, when teams did the final stage of CIDER (`REPEAT`), they simply chose a second assumption from their own list to ideate on, rather than using someone else's assumption.

- Unlike the individual CIDER activities, I did not enforce a time limit on the collaborative CIDER activity, because I felt doing so might unfairly disadvantage students whose teams were spread across multiple time zones or who otherwise found it difficult to meet synchronously during a remote learning quarter.

These collaborative activities additionally contained five open-ended questions for teams about the role of collaboration in their process for the CIDER activity (see Section 5.4.2), as well as asking them to reflect on the experience they gained over the course of the term and some final thoughts on the activity's usefulness.

5.4.2 Data Collection and Analysis

Three researchers participated in data collection and analysis:

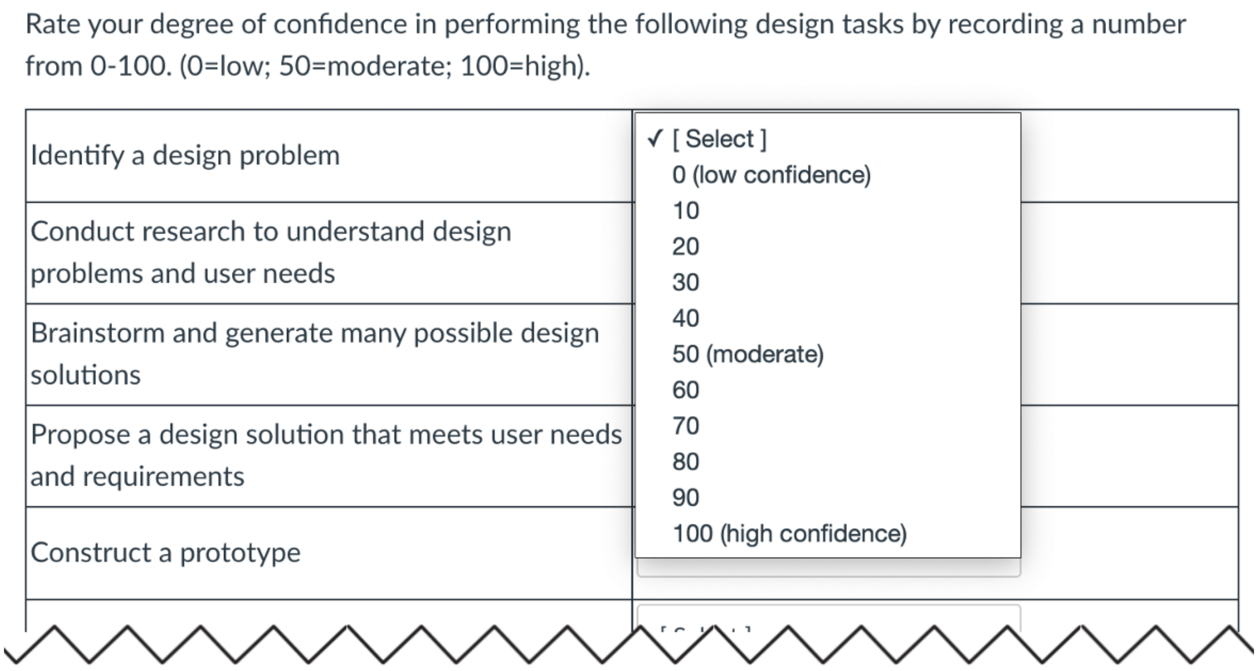
- The dissertation author, a computing education researcher with six years of research experience in inclusive software interface design methods at the time of the study, including three years researching HCI education within that space and half a year of teaching experience in post-secondary computing contexts. The dissertation author was also the instructor for the course in which the case study took place. They conducted the statistical analyses of students' self-efficacy for RQ1 in addition to participating in the collaborative qualitative analyses for RQ2, RQ3, and RQ4.

- The second analyst, a research assistant with two years of research experience in computing education and design methods as well as two years of UX design experience. The second analyst had expertise in qualitative methods and interviewing, conducting the post-class interviews for RQ4 as well as participating in the collaborative qualitative analyses for RQ2, RQ3, and RQ4.

- The third analyst, a research assistant with four years of experience in inclusive software interface design methods, including two years researching HCI education within that context. The third analyst had expertise in qualitative methods and participated in the collaborative qualitative analyses for RQ2 and RQ3.

RQ1: How might CIDER-based activities impact students' self-efficacy as a designer?

At the end of each week of instruction during the quarter, students filled out a short weekly check-in survey. The purpose of these surveys were twofold: First, to be a communication medium through which students could give feedback and raise concerns to the instructor during a period of semi-synchronous remote learning; and second, to capture changes in students' design self-efficacy over time. Within the context of the course, these surveys were graded based on completion (i.e., students received full participation points so long as they logged in and submitted a survey). To


Rate your degree of confidence in performing the following design tasks by recording a number from 0-100. (0=low; 50=moderate; 100=high).

Identify a design problem	✓ [Select]
Conduct research to understand design problems and user needs	
Brainstorm and generate many possible design solutions	
Propose a design solution that meets user needs and requirements	
Construct a prototype	

Figure 5.9: The format of the self-efficacy items as students saw on their weekly check-in surveys.

gather self-efficacy information, I asked students to rate their degree of confidence in performing nine general design tasks and four inclusive design tasks on a scale of 0 (low confidence) to 100 (high confidence) in intervals of 10. The general design items and the scale of measurement were adapted from Carberry et al.'s investigation of engineering design self-efficacy [47]. The inclusive design items were created by the research team to correspond to design tasks carried out when using the CIDER technique, with language and structure mirroring those of the general design items from Carberry et al.'s study. Figure 5.9 shows an example of how the survey presented the self-efficacy items to students, and Table 5.2 lists the text of all 13 self-efficacy items. The result of this was 10 sets of ordinal measures of students' self-reported design self-efficacy for 13 design tasks.

To understand at a high level how students' confidence in their abilities may have changed over time, I conducted nonparametric Wilcoxon U Signed-Rank tests to understand the changes in

General Design Items (adapted from Carberry et al. [47])	
identify-problem	Identify a design problem
conduct-research	Conduct research to understand design problems and user needs
brainstorm-general	Brainstorm and generate many possible design solutions
propose-solution	Propose a design solution that meets user needs and requirements
construct-prototype	Construct a prototype
evaluate-test	Evaluate and test a design
critique-design	Critique a design
iterate-update	Iteratively incorporate feedback and update a design
communicate-design	Communicate about a design
Inclusive Design Items (corresponding to CIDER technique aspects)	
identify-assumptions	Identify assumptions a design makes about users' abilities or contexts
identify-exclusion	Identify ways in which a design might exclude certain types of users
exclusion-scenario	Write a scenario in which a user might not be able to use a design due to an assumption
brainstorm-inclusive	Brainstorm changes to a design that might make it more inclusive

Table 5.2: The thirteen self-efficacy items we asked students on each weekly check-in survey. Students self-reported a score for each item on a scale from 0 (no confidence in their ability to perform the task) to 100 (high confidence) in intervals of ten, as shown in Figure 5.9. Tags in the leftmost column are used to represent each item in the Results section.

general and inclusive design self-efficacy between the beginning (week 1) and end (week 10) of the academic term[5]. I opted to conduct this analysis at the granularity of each skill due to my desire to understand if particular design skill self-efficacies were more or less impacted over the course of instruction. Consistent with the recommendations of the Transparent Statistics in HCI working group [285], I also calculated and report effect sizes of statistically significant results, including the Vargha and Delaney A effect size [290] which provides a means of making a common-language comparison between two groups.

[5]One student did not report self-efficacy scores for week 1, so I used their week 2 scores instead. Similarly, one other student did not report week 10 scores, so I used their week 9 scores instead.

To supplement these statistical analyses, for each weekly survey after the first, I asked students to self-report their perceptions of whether they were *more confident*, *about the same*, or *less confident* in their ability to do general and inclusive design work compared to the previous week, and if they perceived a change, what they thought led to that change. This provided us with explanatory qualitative data to help us interpret any changes in students' reported self-efficacy scores. To analyze the qualitative feedback gathered on these open-ended items, one researcher (the second analyst) conducted a thematic analysis [235] with a sensitizing concept of *becoming more or less confident in design skills*. The results of this analysis were iteratively shared and discussed with the rest of the research team until collaborative agreement on the major themes was achieved.

Finally, on the Week 6 check-in survey, I also asked students to rank the contributions of different components of instruction to their personal design learning. Week 6 represented the "halfway" point of the course, where students pivoted from completing readings and discussions on their own to working on their final design projects (see Table 5.1). I opted to ask for students' ranking of course components in Week 6 rather than the end of the term in order to capture this information when it was more immediate in students' minds. The aspects of the course I asked students to rank included:

- *Required readings*, such as chapters from the course's textbook;
- *Optional readings*, which were supplemental to the required readings;
- *Reading quizzes*, single-question comprehension check quizzes based on the required readings;
- *Peer discussion boards*, where students asked and answered questions about readings;
- *Deliverables*, weekly activities where students practiced design skills with a partner;
- Optional *synchronous discussion sections*, where students could (virtually) discuss design topics with peers and the instructor;

- and the individual *CIDER activities*, which students completed weekly from Weeks 2-6.

I also included a “something else” response in the ranking options where students could fill in their own answer (such as an internship or hackathon they had participated in). I asked students to assign each of these options a rank from 1 to 8, with 1 indicating *“I learned the most from this”* and 8 indicating *“I learned the least from this”*, and to elaborate on their rankings as much as they wished in an open-ended response. To analyze these rankings, one researcher (the second analyst) examined students’ responses for trends, noting for each course component how many students had ranked it as one of the most helpful to their design learning and what students mentioned about it in their open-ended responses. Given the nature of the study, the researcher focused in particular on any trends in students’ rankings of and comments about the CIDER activities. These results were shared and discussed with the rest of the research team, who collaboratively came to agreement on the nature of the observed trends and their potential interactions with student self-efficacy.

RQ2: How might the CIDER technique help students recognize different types of exclusionary design biases?

For each of the five individual CIDER activities performed by students in weeks 2-6 of the course, for each student, I collected a list of assumptions they identified during the technique’s `CRITIQUE` stage; two scenarios of exclusion (one from the `IMAGINE` stage and one from the analogous part of the `REPEAT` stage); and two lists of proposed redesign ideas to make the design more inclusive (one from `DESIGN` and one from `REPEAT`).

To analyze how students’ recognition of different types of design bias may have changed over time, I began by categorizing the types of assumptions students came up through iterative inductive coding. Across the five individual CIDER activities, I collected 1259 student-generated assumptions about users. Two researchers (the second and third analysts) collaboratively affinity diagrammed a subset of the assumption data in order to generate initial themes for the coding efforts, memoizing their rationale as they developed a set of categories which fit the data well. Once they felt they had a stable set of themes, they shared and discussed the categories with the other member of the research team (the dissertation author), adjusting the codeset as needed until all three researchers agreed on the categories and their descriptions. All researchers then divided and coded the remaining assumptions by type according to the agreed-upon codeset, recording rationale for their coding when appropriate. I allowed for multiple codes to be applied to each assumption item during this coding effort, because it was possible (though uncommon) for one assumption statement to identify multiple different types of embedded assumptions. After this process was finished, the research team met once more to review the results of the coding effort and discuss any discrepancies in the application of the codeset, collaboratively adjusting the coded data as needed after discussion of interpretations and reaching agreement on the major types of assumption present.

In addition to the above, I analyzed students' coverage of the major assumption types across their individual CIDER activities. I did this to better understand whether students really were able to surface new types of assumptions over time, or whether they simply repeated the same types of assumptions they had mentioned before. If students mentioned increasingly more types of assumptions over time, it could signify their gaining new perspectives on different ways designs could exclude potential users, thus building their knowledge bases of design bias examples for future design work. One researcher (the dissertation author) wrote a script which operated on the assumptions lists students produced during their CIDER activities, which were coded by type as described in the previous paragraph. For each student, the script output their cumulative coverage of identified assumption types, broken down by week to give a sense of potential change over time. Combined with quotes from students' weekly check-in surveys, this enabled us to understand how students' abilities to recognize different types of design bias may have changed, as well as what may have led to those new understandings.

Though I collected 2,246 student-generated redesign proposals from the individual CIDER activities' `DESIGN` stages, I opted not to analyze these other than to note that every student was able to generate at least one redesign idea to make a design more inclusive for each CIDER activity they completed (minimum=1, maximum=13, median=5). Given the nature of this case study and its situatedness within the broader course, it would be difficult to disentangle the effect of the

CIDER from the effects of general instruction and/or prior knowledge on the type or number of redesigns students were able to propose. Instead, I chose to focus on the more salient part of the intervention: students' abilities to identify different kinds of design bias, and how the CIDER activities may have improved that recognition over time. Future work around this technique should investigate the influence of the CIDER technique on the types and numbers of inclusive redesigns students come up with.

RQ3: How might conducting CIDER-based activities collaboratively, rather than individually, impact students' experiences?

Fifteen teams completed a collaborative CIDER activity using their own designs as artifacts of analysis. Each team produced a single list of assumptions about users which they identified in their design during the `CRITIQUE` stage, chose two of those assumptions to focus on in sequence, and then provided a scenario of exclusion (`IMAGINE` stage) and a brainstormed list of redesign ideas (`DESIGN` stage) for each target assumption.

To understand teams' experiences during the collaborative CIDER activity, I asked them to respond to five open-ended reflection questions. These questions were included at the end of the activity and were answered after teams completed their collaborative CIDERs. The questions as presented to students were:

1. *Previously, we did these activities alongside our textbook readings and discussions. Now, you've had a chance to practice and apply your design knowledge through project work. Did the experience you gained over the past few weeks change your approach or the kinds of responses you gave to this activity? Why and how, or why not?*

2. *What was different about doing this activity on your own design rather than someone else's designed artifact?*

3. *What was different about doing this activity with teammates rather than individually?*

4. *Do you think this activity helped you uncover meaningful limitations of your design? Why or why not?*

5. *If you were to continue working on this project beyond the end of the quarter, do you think it would be feasible to address all (or most of) the assumptions you uncovered? Would there be any unavoidable tradeoffs? Explain your thinking.*

To answer RQ3, two researchers (the first and second analysts) conducted collaborative qualitative thematic analyses on teams' responses to each of the five reflection questions. The two researchers used initial sensitizing concepts of *differences between team and individual contexts*, *using CIDER on one's own design* and *perceptions of the activity's usefulness in team contexts* depending on the question. The two researchers shared and discussed the results of their analyses with the third analyst, and together the research team came to agreement on major trends which arose from students' responses related to collaboration, teamwork, and how these themes interacted with teams' usage of the CIDER technique.

RQ4: What kinds of lasting impacts might the CIDER technique have on students' design approaches?

To better understand how the use of the CIDER technique might have influenced students' perspectives on design and inclusion, I conducted semi-structured interviews with students from the class after the term was over. To recruit for these interviews, on the final weekly check-in survey, I included an item that asked students to leave their email address if they were interested in participating in a short, compensated follow-up interview about their experiences in the course after the term had concluded. The item on the survey included a note to students that participation in these interviews was entirely optional and would have no impact on their grades or any other course outcomes, because the instructor (the dissertation author and lead researcher) would not see their responses until after final grades had been submitted. To ensure this held true, the instructor did not participate in analysis of the final week's survey data until after they submitted final grades and resolved any marking discrepancies.

17 students left their contact information on the final survey to indicate interest in participating in interviews. The instructor sent an initial recruitment email to these students three weeks after the conclusion of the quarter. This email contained more information about the goal of the interviews (i.e., to understand students' experiences in the course, specifically around the CIDER activities) and offered participants a $10 gift card to participate in a half-hour interview. To account for preferences and availability, I offered students the option of participating in the interviews over a remote video call or through email. In an attempt to avoid biasing participant responses, all communication after the initial email, including the interviews themselves, was carried out by the second analyst, a researcher who was not involved in or connected to the course itself. Six students responded to the recruitment email and agreed to participate in the post-class interviews.

The interviews themselves were semi-structured and carried out by the second analyst. The content of the interviews focused on understanding students' experiences with design before, during, and after the class, with a particular focus on how their perspectives on design and inclusion had shifted during the course, if at all. The interviewer also asked students to tell them about their experiences with the course activities based on the CIDER technique, such as whether and how those activities had played a part in shaping any newfound perspectives on design and inclusion. Finally, the interviewer asked students to tell them their biggest takeaways from the course overall, and if they were comfortable sharing, to describe how those takeaways had impacted any design work they had done since the course concluded (e.g. if they were on an internship or working on personal portfolio projects). The interviews ended with a general open question *"Is there anything else you'd like to share with us?"* to enable participants to fully share their thoughts and experiences as much as they wanted.

Interviews were recorded and transcribed for analysis if they took place over video call. If the interviews were conducted over email, I used the text of the email responses as the source of data. To analyze these transcripts, one researcher (the second analyst) conducted a thematic analysis on the responses with sensitizing concepts of *shifts in perspectives on design* and *newfound understandings of inclusion*, noting in particular places where students mentioned impacts of the CIDER activities on their learning or design approaches. Then, the dissertation author and the second analyst discussed the results of this analysis, returning to the data when necessary to collaboratively converge on agreement about the major themes that arose from students' narratives.

5.5 Case Study: Results

5.5.1 RQ1: How might CIDER-based activities impact students' self-efficacy as a designer?

Overall, more than half the students in the class (22/40) ranked the CIDER activities as one of the top three aspects of the class that helped them learn the most, with almost a third of students (13/40) ranking it within the top two, and a fifth (8/40) as the most conducive to their overall design learning. For context, students only ranked two other aspects of the class consistently higher with regards to how much they learned: the required readings (36/40 top three, 31/40 top two, 19/40 most helpful), which formed the core of class instruction, and the weekly deliverables (26/40 top three, 24/40 top two, 11/40 most helpful), where they practiced and applied design skills.

Figure 5.10 shows the histograms of students' self efficacy scores and Table 5.3 shows results of my nonparametric analyses for each of the 13 design skills I asked about in the weekly surveys. Differences between the median self-efficacy scores for all skills were statistically significant according to Wilcoxon U Signed-Rank tests. For each skill, I also calculated effect size according to the guidelines from the Transparent Statistics in HCI working group [285]. I identified a large effect size for all skills (Table 5.3, "Effect Size"). Finally, I calculated the Vargha and Delaney A effect size [290] for a common language comparison between the two groups, which can be interpreted as a probability. I interpret the large Vargha and Delaney A effect sizes for each skill to state that there is between an 81-91% chance that the self-efficacy score for a random student from week 10 will be higher than the self-efficacy score for a random student from week 1, depending on the skill in question (see rightmost column in Table 5.3).

A rise in design self-efficacy is to be expected given the introductory nature of the course: Students with little-to-no design experience *should* feel more confident after practicing design skills for several weeks. To understand how CIDER activities may have influenced students' self-efficacy gains, I turned to the qualitative data collected on weekly surveys. Several students with varying

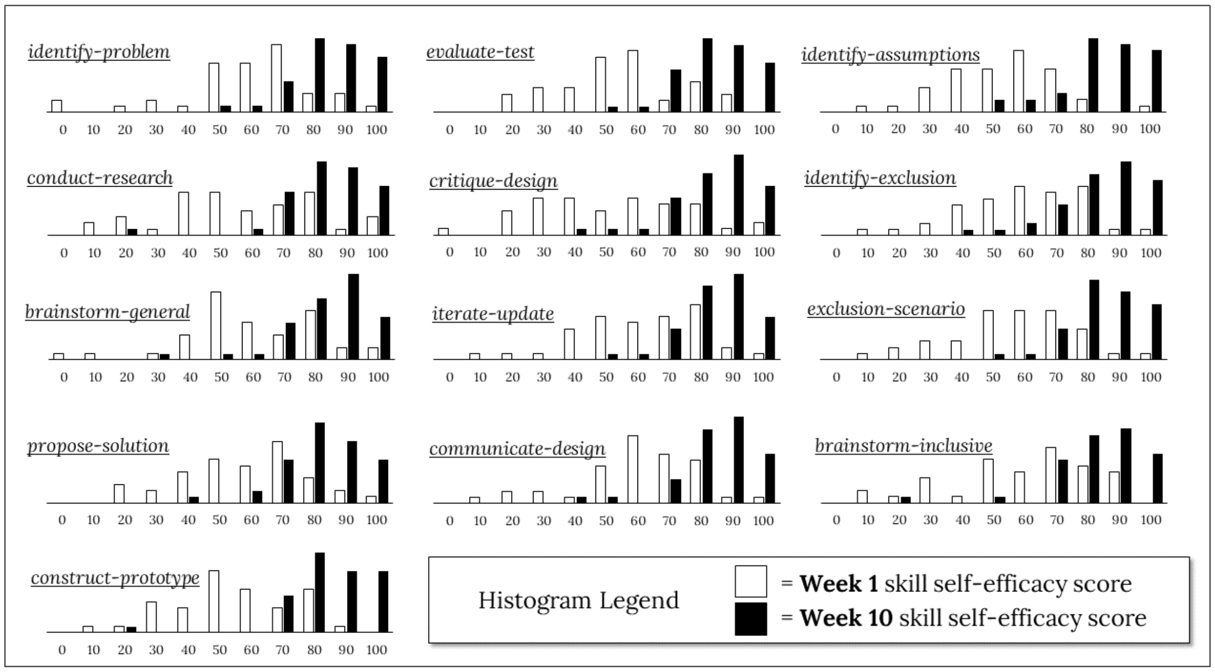


Figure 5.10: Distributions of students' self-efficacy scores on 13 design skills, comparing students' beginning (Week 1) and ending (Week 10) self-efficacy ratings. Skill tags correspond to those listed in Table 5.2.

self-efficacy trajectories mentioned CIDER activities in their open responses. Some students began the course with very little confidence in their inclusive design skills, but showed large gains in confidence over the term. For instance, P10 started the course with the lowest self-reported inclusive design self-efficacy of the class, with a median score of 10 on four inclusive design skills in the Week 1 survey. They took a few weeks to report feeling more confident about inclusive design, citing a lack of familiarity with many of the skills they were practicing, but during Week 5, they were able to see their own improvement through their answers on the CIDER activity:

> P10, Week 5 survey: *"This week helped me put more principles into action ... the ability to visibly see my improvement through the exercises [CIDER activities] this week was awesome - really helped me feel more confident with my ability."*

	Week 1 SE		Week 10 SE		Wilcoxon U Signed-Rank		Effect Size	Vargha and Delaney A
Design Skill	**Median**	**IQR**	**Median**	**IQR**	**Z**	**p**	**r = Z/sqrt(N)**	**A**
identify-problem	60	20	90	12.5	5.1255	<.0001	0.8104	0.8700 (large)
conduct-research	55	40	80	10	4.9255	<.0001	0.7788	0.8191 (large)
brainstorm-solutions	60	30	90	10	4.9043	<.0001	0.7754	0.8144 (large)
propose-solution	60	22.5	80	12.5	5.2605	<.0001	0.8318	0.8469 (large)
construct-prototype	50	30	85	12.5	5.214	<.0001	0.8244	0.8988 (large)
evaluate-test	55	22.5	80	10	5.3876	<.0001	0.8519	0.8844 (large)
critique-design	50	40	90	10	5.3535	<.0001	0.8465	0.8616 (large)
iterate-update	60	30	90	10	5.0855	<.0001	0.8046	0.8441 (large)
communicate-design	60	30	90	10	4.9819	<.0001	0.7878	0.8453 (large)
identify-assumptions	55	22.5	90	12.5	5.4449	<.0001	0.8609	0.9147 (large)
identify-excluded	60	22.5	90	10	4.9713	<.0001	0.7860	0.8416 (large)
write-scenario	60	20	85	10	5.4368	<.0001	0.8596	0.8866 (large)
brainstorm-inclusive	65	30	85	10	4.9089	<.0001	0.7762	0.8172 (large)

Table 5.3: Results of nonparametric statistical analyses of students' self-efficacy scores on thirteen design skills (nine general and four specifically related to designing inclusively with CIDER). Tags in the *Design Skill* column correspond to those listed in Table 5.2.

The repeated CIDER activities seemed to give P10 a mechanism to reflect upon and see concrete gains in their ability to design inclusively, contributing to increased confidence. By the end of the course, P10 reported a median self-efficacy score of 85 on the four inclusive design skills. For students like P10 who came into the class with little confidence in their ability to design inclusively, the concrete examples and feedback provided by repeated CIDER activities may have helped them better understand and practice inclusive design.

Conversely, some students reported that initial engagement with CIDER activities had *decreased* their confidence in being able to do inclusive design, because they suddenly realized just how much they had previously been missing or overlooking. For instance, P13 reported a median self-efficacy score of 40 across the four inclusive design skills during Week 1, which dropped to a median of 35 in Week 2. They ascribed this drop to the CIDER activity in their survey response:

> P13, Week 2 survey: *"I feel slightly less confident in my inclusive design skills again because this week's activities showed me that I struggled a little more than I thought I would with identifying assumptions and generating viable solutions. This change in confidence is not a result of this class or its structure, it is the result of a reality check the inclusive design activities gave me"*

However, P13 quickly bounced back and gained confidence in their inclusive design skills over the next few weeks. By Week 6, the week of the final individual CIDER activity, they reported a median inclusive design self-efficacy score of 80, which further rose to a median of 90 by Week 10. They reported feeling more and more confident in their ability to do inclusive design over time, ascribing the gains to the practice they got from repeating the inclusive design activity on different artifacts, as well as the experience they gained from their final project work and having to consider inclusion from the perspective of a practicing designer rather than only as a critic. For these types of students, the CIDER activities may have played a part in showing them just how much they still had to learn about inclusion, dispelling the notion that good design is easy to achieve and making them more mindful of areas they could improve.

Students with prior design experience and high inclusive design self-efficacy at the beginning of the course rarely reported that CIDER activities influenced their self-efficacies, though they seemed to find the activities useful for reinforcing existing competencies. For instance, P28 self-reported an average inclusive design self-efficacy score of 97.5 across the four skills I measured in Week 1 of the course, attributing their confidence to past internships and design projects they had led or participated in, as well as a past project they had done specifically on inclusive design. P28's confidence in their inclusive design skills remained high throughout the course, though they did note a few ways that the CIDER activities helped them reflect upon challenges they faced in their own design processes:

> P28, Week 3 survey: *"If inclusive design is designing for as many users as possible and taking into account the needs and abilities of as many users as possible then I would think [I have] the same [level of confidence as last week] because in some cases*

Code	Description: Assumptions about a user's...
Prior Knowledge (PK)	Knowledge of affordances or familiarity with similar interfaces/interaction styles
Vision (Vi)	Level, extent, and/or type of visual ability
Hearing (He)	Level, extent, and/or type of hearing ability
Motor (Mo)	Level, extent, and/or type of motor ability
Cognition (Co)	Level, extent and/or type of cognitive ability
User Context (UC)	Physical, social, or cultural context of the user
Access to Technology (AT)	Access to technology and/or resources needed to engage with it (Internet, electricity)
Device Specifications (DS)	Device capabilities (speed, processing power) and/or hardware (inputs, outputs)
Language and Literacy (LL)	Fluency and/or literacy level with a particular language

Table 5.4: The nine types of assumptions students identified throughout their individual CIDER activities, including assumptions about different kinds of user capabilities as well as assumptions about users' broader contexts.

it becomes hard for me to identify what to do unless I see their experiences."

P28 did rank the CIDER activities as one of the parts of the class that was most helpful to their learning in Week 6, tied for first place with the weekly deliverable assignments, indicating they did receive some value from the activities. Their comments on later surveys ascribed this value largely to the repeated practice with the format, noting that they were getting quicker at listing assumptions. For students with existing in-depth inclusive design knowledge, CIDER activities may be better for practicing and reinforcing existing inclusive design skills rather than necessarily imparting new perspectives on design.

5.5.2 RQ2: How might the CIDER technique help students recognize different types of exclusionary design biases?

Students identified assumptions about users' prior knowledge, capabilities, and broader contexts.

Table 5.4 describes the nine types of assumptions about users students identified which emerged from my analysis of students' `CRITIQUE` lists on the individual CIDER activities. There were also three categories of responses from students which did not contain assumptions: statements about user preferences (e.g. *"The user might not like the color scheme"*), general critiques of the interface which did not contain assumptions (e.g. *"This looks more like a blog than a table of contents"*), and statements which were incomplete, incomprehensible, or otherwise did not fit into the assumption categories, which were marked with an "other" code. Students' non-assumption responses are not reported on in this chapter.

For this analysis, I adhere to the perspective on qualitative coding proposed by Hammer and Berland [132], in which I treat the results of the coding effort as organizations of claims about data, rather than quantitative data in and of itself. As a result, I do not report exact code frequencies in the following subsection or calculate metrics such as inter-rater reliability, preferring instead to focus on characteristic descriptions of code instances I observed in the data. Below, I present examples of each assumption-related category and code below, supported by quotes from students' CIDER activities.

Some assumptions students identified revolved around concerns that are generally considered in traditional design processes, even if accessibility and inclusion is not a focus. When students identified these kinds of assumptions, they highlighted how the interfaces or interaction styles of the designs under scrutiny relied on some facet of a user's *Prior Knowledge (PK)* to work correctly. For instance, multiple students mentioned that several designs relied on the user to have prior experience with similar kinds of technology, especially when there were no easily evident indicators to help first-time users:

> P40, Week 6: *"Touchscreen-only interface assumes users have a basic understanding of how to use a touchscreen"*

Others specifically mentioned the reliance on the user's recognition of common affordances [135], especially after the fourth week of the term, when the course covered concepts such as gulfs of execution and evaluation [148]:

> P3, Week 4: *"Usage of blue links assumes that users are familiar with the understanding that hyperlinks are underlined and blue"*

PK codes were some of the most commonly identified types of assumptions, with at least two-thirds of the students listing one or more *PK* assumptions each time I conducted a CIDER activity (see Table 5.5).

When reflecting on the inclusiveness of a design, many students also identified assumptions about users which had to do with a their potential physical or mental capabilities. Assumptions related to a user's potential *Vision (Vi)* manifested when students identified potential bias around a design's sole reliance on visible cues to convey information:

> P16, Week 2: *"Aside from the two bumps on the 'F' and 'J' key, keyboard assumes that user has the ability of sight and can determine where certain keys are"*

A common *Vi*-coded assumption I observed was that of colorblindness, in which students often pointed out that not all users could distinguish all colors easily:

> P19, Week 3: *"The different color of the orange "New Meeting" icon compared to the blue icons assume the user is not color blind and call tell the difference"*

Students only identified assumptions related to a user's potential *Hearing (He)* ability in the Week 3 (Zoom) and Week 5 (Google Home) CIDER activities, likely due to the salience of audio-based interactions in these artifacts' designs. Students who surfaced *He*-coded assumptions listed different ways that a design might be biased against users who could not hear or process audible information in the particular way that the design conveyed it:

> P22, Week 3: *"Users are not deaf and are able to hear other users."*

> P29, Week 5: *"assumes that the user can hear or at least hear the frequency at which the Google Home talks"*

Sometimes, students identified *Motility (Mo)*-based assumptions that designs made about users. These were more prevalent when the physical interactions required to use a design were more salient, such as how a keyboard and mouse (Week 2) respond largely only to tactile input, or the toaster (Week 6) having only touchscreen-based interactions:

> P2, Week 2: *"There are a number of edge cases which may provide difficulty for some people if they are blind or have difficulty with some finger movements (ex. arthritis)."*

> P29, Week 6: *"assumes the user has enough motor control to accurately choose options on the small touch interface"*

Students occasionally surfaced assumptions embedded into designs which were related to a user's potential *Cognition (Co)* or cognitive abilties. Overall, relatively few students mentioned *Co*-coded assumptions each week (see Table 5.5). This is consistent with prior work reporting that neurodiversity is an often-overlooked facet of inclusion in the design of technology [70]. However, the handful of students who did surface *Co* assumptions noted a breadth of ways designs might be biased against neurodivergent individuals:

> P25, Week 4: *"is the font dyslexic friendly? (or other condition friendly)"*

> P28, Week 5: *"a person with color-taste synesthesia might feel a bad taste if one of the colors is associated with a taste for them"*

The *Vi*, *He*, *Mo*, and *Co* assumption types signify students' recognition of a number of ways that artifacts might perpetuate exclusion when their designs are built on assumptions about users' physical or mental capabilities. Accessibility and ability-related topics like these are often included in inclusive design resources (such as [5, 210]), though as discussed in the Related Works (Section 5.2), it can be difficult to get students to recognize and act upon them in their own design practice [192]. The fact that these kinds of ability-related assumptions were consistently identified by students across the five individual CIDER activities performed indicates promise for the technique's ability to help students surface different kinds of user diversity.

Sometimes students uncovered assumptions which went beyond a users' potential ability to their surrounding contexts and resources. When students identified embedded assumptions that had to do with a user's environment, I categorized these as *User Context (UC)* codes. For instance, some students highlighted how cultural or societal contexts might influence a user's ability to interact with a design:

> P9, Week 2: *"The dollar sign on the keyboard tells us that the product is centered towards the sales in the United States."*

Others identified ways in which designs made assumptions about the properties of a user's physical location:

> P29, Week 4: *"Doesn't give the time zone in which the page was updated so assumes that users are either using PST or assuming PST when visiting the page"*

> P3, Week 5: *"Primary mode of communication being voice assumes that users have a quiet enough space to place the Google Home where it can hear the user's voice"*

Students also identified instances in which designs assumed particular levels of *Access to Technology (AT)* that users might or might not have. Notably, *AT*-coded assumptions were often acute barriers that might have entirely prevented a user from interacting with the design as intended. Several touched on issues that might have been outside the designers' control, yet were important to consider in light of device use, such as the assumption that users would have reliable Internet or electricity:

> P23, Week 3: *"Users have access to reliable, high-speed internet to hear + see everything live-time"*

> P39, Week 6: *"It assumes that users have access to electricity in order to turn on the toaster"*

Assumptions coded as relating to *Device Specifications (DS)* were similar in some ways to the previously mentioned *UC* and *AT* codes, but they had to do in particular with the capabilities, inputs, and outputs of devices that supported the HCI artifact's use. As can be seen in Table 5.5, students surfaced these kinds of assumptions more often when the target artifact of analysis was a piece of software (e.g. Zoom) or a hardware device that connected to other devices (e.g. the keyboard and mouse), and rarely or never when the artifact was self-contained (e.g. the non-networked touchscreen toaster).

> P27, Week 2: *“Keyboard has a Windows symbol so it assumes it will be used with a machine running Windows, and with the specific Windows symbol it assumes Windows 8 or later”*

> P2, Week 4: *“Assumes the device has a large enough resolution to display everything on screen”*

A final category of assumptions students identified were those related to users’ potential literacy levels and their fluency in various languages, which I coded as *Language and Literacy (LL)* assumptions. These sorts of assumptions arose more often when the design’s interactions relied heavily on text, such as the COVID-19 information webpage or the touchscreen toaster’s interface, or spoken audio, such as the Google Home voice assistant.

> P25, Week 4: *“users understands English very well (part of [university] is the [program for int’l. students] and I know for a fact that some of them doesn’t have a fluent understanding of the language which is why they are here to learn and improve their English.)”*

> P11, Week 6: *“assumes the user can read English “Caution: Hot surface” and the “Face bagel inward” labels”*

Students’ identification of *UC*, *AT*, *DS*, and *LL*-related assumptions indicate understandings of inclusion that can extend beyond a user’s direct interaction with an interface and encompass their surrounding contexts. Cultural and social contexts, access to reliable technology and resources, and literacy and fluency levels can all substantially impact a person’s ability to interact with a design, though these facets are not always considered in technology design processes [70]. Engaging with peers’ assumptions about broader inclusion during the CIDER `EXPAND` stage may help students become more aware of the multiple ways users’ contexts impact their interactions, leading to expanded understandings of how designers’ assumptions can be fundamentally incompatible with users’ realities.

CIDER Activity	PK	Vi	He	Mo	Co	UC	AT	DS	LL
Week 2 (keyboard/mouse)	28 (70%)	28 (70%)	0 (0%)	29 (73%)	1 (3%)	12 (30%)	4 (10%)	17 (43%)	10 (25%)
Week 3 (Zoom)	28 (70%)	18 (45%)	13 (33%)	8 (20%)	3 (8%)	14 (35%)	30 (75%)	26 (65%)	17 (43%)
Week 4 (COVID info site)	27 (68%)	15 (38%)	0 (0%)	3 (8%)	5 (13%)	20 (50%)	18 (45%)	13 (33%)	23 (58%)
Week 5 (Google Home)	34 (85%)	14 (35%)	25 (62.5%)	32 (80%)	3 (8%)	13 (33%)	19 (48%)	5 (13%)	12 (30%)
Week 6 (Touchscreen toaster)	36 (90%)	29 (73%)	0 (0%)	23 (58%)	1 (3%)	20 (50%)	10 (25%)	0 (0%)	23 (58%)

Table 5.5: Count and proportion of how many students identified at least one assumption of a given type during the `CRITIQUE` stage of each CIDER activity. Percentages given are proportions out of 40 students, rounded to the nearest whole number. PK=Prior knowledge, Vi=Vision, He=Hearing, Mo=Motor, Co=Cognition, UC=User context, AT=Access to technology, DS=Device specifications, LL=Language and Literacy.

Students tended to surface different types of assumptions when critiquing different artifacts

To explore patterns and trends in the types of assumptions that students identified across the five individual CIDER activities, I counted the number of students who identified each type of assumption at least once in a given CIDER activity's `CRITIQUE` stage. Table 5.5 shows these results. Some types of assumptions were identified by most students across all activities—For instance, more than two-thirds of students identified at least one *Prior Knowledge (PK)*-coded assumptions in the `CRITIQUE` stage of every CIDER activity, and sometimes that proportion was as high as 90% (see Table 5.5, column 2). Other types of assumptions were less consistently surfaced. Students identified *Hearing (He)*-coded assumptions only in the CIDER activities corresponding to weeks 3 and 5, where the artifacts of analysis were Zoom and a Google Home digital assistant, respectively (Table 5.5, column 4). During week 6, when the artifact of analysis was the toaster with the touchscreen-based interface, no students noted assumptions having to do with users' *Device Specifications (DS)* (Table 5.5, column 9, last row).

These trends suggest that the choice of artifact for each CIDER activity, and specifically the salient interaction paradigm of the design under critique, influences the kinds of design bias students might notice. When the Week 3 (Zoom) and Week 5 (Google Home) CIDER activities made

CIDER Activity	PK	Vi	He	Mo	Co	UC	AT	DS	LL
Week 2 (keyboard/mouse)	28	28	0	29	1	12	4	17	10
Week 3 (Zoom)	+9	+8	+13	+1	+3	+10	+28	+14	+13
Week 4 (COVID info site)	+3	*nc*	*nc*	*nc*	+4	+11	+2	+1	+8
Week 5 (Google Home)	*nc*	+2	+15	+8	+2	+1	+2	+1	+1
Week 6 (Touchscreen toaster)	*nc*	*nc*	*nc*	*nc*	+1	+4	*nc*	*nc*	+3
Totals	40 (100%)	38 (95%)	28 (70%)	38 (95%)	11 (27%)	38 (95%)	36 (90%)	33 (83%)	35 (88%)

Table 5.6: Cumulative count of how many students mentioned an assumption of that type for the first time during the assumption generation part of CIDER, using Week 2's initial count as a baseline. Cells with "nc" indicate there was no change since the previous week; i.e. no students mentioned that code for the first time during that week's activity. Percentages at the bottom are out of 40 students, rounded to the nearest whole number, and represent the proportion of students who had mentioned that type of assumption at least once by the final individual CIDER activity.

audio interactions salient due to the artifacts' reliance on speech and sound, students considered and successfully identified *Hearing (He)*-coded assumptions, but not otherwise. Similarly, when students were asked to consider a self-contained device with no obvious networked connections or outputs in the Week 6 touchscreen toaster activity, they did not consider assumptions about a user's *Device Specifications (DS)* might impact their experience with the design – likely making the assumption themselves that the designer of the toaster made it suitably powerful enough to do what it was meant to do. Certain types of assumptions might be surfaced more or less often depending on the object of critique during CIDER activities, which suggests that educators should take care to represent a diversity of interaction styles and device types when picking their artifacts of analysis for use with the technique.

Students showed increasing awareness of different kinds of assumptions with each subsequent CIDER activity

A key goal of the CIDER technique is that its use should help students expand their knowledge bases of design bias by providing concrete examples of exclusion, enabling them to recognize more types of design bias than they could before. If we consider the nine coded types of assumptions from the previous subsections to make up the possible "assumption space" within which students might identify specific manifestations of design bias, students then would signal broadening understandings of the assumption space when they identify a new type of assumption on their CIDER activities. To better understand how students demonstrated broader understandings of assumption spaces, I analyzed students' sets of assumptions they created during the `CRITIQUE` stages of the CIDER activities, and looked at how many students were identifying new types of assumptions each week (i.e. coded assumption types they had not yet mentioned on previous CIDERs). Table 5.6 shows the results of this analysis, including the final number of students who identified an assumption of each given type by the final individual CIDER activity.

Over time, the CIDER activities seem to have helped students recognize and identify increasingly many different kinds of embedded assumptions which could lead to bias and exclusion. By the final CIDER activity, nearly all students (90%+) had successfully identified assumptions relating to users' potential *Prior Knowledge (PK)*, *Visual (Vi)* and *Motor (Mo)* ability, surrounding *User Context (UC)*, and their *Access to Technology (AT)*. Many students (70%+) also identified at least one assumption relating to users' potential *Hearing (He)* ability, their *Device Specifications (DS)*, or their *Language and Literacy (LL)* fluency. The least-identified type of assumption had to do with users' potential *Cognitive (Co)* abilities, but even then, slightly more than a quarter of the students in the class were able to identify at least one assumption of this type.

Compared to the baseline of the very first CIDER activity I conducted, more students showed recognition all of 9 coded types of assumptions after the final activity (Table 5.6, *Totals* row) than they did on the first activity (Table 5.6, Week 2). This is a strong indicator that students' understandings of the assumption space broadened over time. Indeed, by the final individual CIDER activity, all students had identified at least two more types of assumptions than they had on the very first activity. While some of this increase can likely be attributed to early designers learning more about design alongside the progression of the class, several students specifically mentioned the ways in which CIDER activities were the catalyst for their consideration of new perspectives. For instance, P29 reported that the CIDER activities helped open their eyes to new perspectives:

> P29, Week 6 Survey: *"The inclusive design activities are really helpful to help me push myself to understand inclusive design more and expand my perspective. They push me out of my comfort zone and getting me thinking in ways I wouldn't have otherwise."*

P8, who reported gaining more confidence in their inclusive design skills on the Week 4 survey, mentioned that the CIDER activities had enabled them to better consider diverse kinds of users:

> P8, Week 4 Survey: *"[I feel] More [confident] because I found the inclusive activities we have done so far to be helpful and to really get me thinking about the diverse range of users that exist and how they are put at a disadvantage."*

P36 reported being more confident on their Week 5 survey, specifically pointing to the socially-sourced list of assumptions created during the `EXPAND` stage as a mechanism for gaining new insights on inclusion:

> P36, Week 5 Survey: *"The [CIDER] inclusivity activity this week was beneficial because it was relevant and fellow classmates posted lots of good assumptions which I was able to get insight from."*

P18 highlighted the benefits of practice with respect to expanding their understanding of inclusion, specifically using the CIDER activity multiple weeks in a row:

> P18, Week 3 Survey: *"We consistently do the same [CIDER] exercise every week during which we are expected to think about how to include more people into the design. Doing this every week makes it easier each week."*

Given the nature of the case study and the integration of CIDER activities into the course under study, it is difficult to isolate the exact contributions of the CIDER technique to students' increased recognition of design bias. However, these quotes from students make it apparent that the CIDER activities played at least some part in helping some students consider perspectives that they would not have had the opportunity to otherwise, and thus did help students build their knowledge bases with new examples of design bias and exclusion.

Notably, I observed the largest increases in students' demonstrations of assumption space coverage during the first few CIDER activities, with diminishing returns in later activities. This is especially evident for the assumptions related to users' broader contexts in particular (see the comparatively larger numbers in the Week 3 row, columns *UC*, *AT*, *DS*, and *LL*, indicating that many students identified these kinds of codes for the first time in Week 3). The sharp increase in the number of students who were able to identify different kinds of assumptions after completing only one or two CIDER activities is promising, because it suggests that the CIDER technique may be beneficial even if only a single instance of it is integrated into a class. Exposure to the kinds of assumptions made visible to students by the CIDER technique's `CRITIQUE` and `EXPAND` stages, even if only experienced once, seems to be effective in helping students recognize more kinds of exclusionary design biases in existing artifacts.

5.5.3 *RQ3: How might conducting CIDER-based activities in project teams, rather than individually, impact students' experiences?*

As mentioned previously, students conducted a collaborative CIDER activity with their project teams in the tenth week of the course. For this activity, they analyzed their own prototypes and worked together to identify and address assumptions embedded in their designs, then answered a few reflection questions on how the team-based experience differed from doing CIDER activities on their own. I consider the collaborative CIDER activity to provide potential insights into how the technique might be used in a more authentic setting, such as if it were to be used in a design evaluation outside of a classroom context. Even though the majority of my students were early designers and the length of teams' design processes were artificially constrained due to the academic term, I still uncovered several interesting insights of how CIDER might support inclusive design work beyond classroom contexts.

Collaborative CIDER activities enabled students to consider more perspectives and seemed more fun

When asked to reflect on how doing the CIDER activity in teams was different than doing it individually, several teams reported it was easier to come up with a breadth of different assumption types during the activity's `CRITIQUE` stage when they had a chance to discuss with teammates:

> Team A: *"Doing this as a group definitely makes uncovering a lot of the design assumptions a lot easier because we pass our ideas around and build on each other. We also get to filter out a lot of ideas that may have been irrelevant."*

Many teams also mentioned the benefits of having multiple designers, each with their own perspectives and experiences, contribute to the same analysis. They often claimed that this allowed them to identify more kinds of assumptions or redesign ideas than they would have been able to individually:

> Team G: *"By having multiple people looking at the design from different perspectives, you end up with a wider range of assumptions. Working on the [assumptions] list as a group also opens your eyes to perspectives other than your own and lead you to notice things you wouldn't have before. "*

> Team N: *"Having different lenses that we use to view the world and different backgrounds helped us [...] we were able to come up with different [redesign] solutions that we would not have come up with if we had been working on our own."*

A few teams found the collaborative CIDER activities more enjoyable than the individual activities, such as Team B, who simply stated:

> Team B: *"This activity is more fun to do with a group!"*

The above insights suggest that collaboration with teammates may be an effective way to help students expand their personal knowledge bases of design bias. In the individual CIDER activities, the technique's `EXPAND` stage played this role by providing a means for students to see and engage with assumptions identified by their peers, who likely had different experiences and background knowledge. In the collaborative activities, coming up with an assumption list alongside teammates during the activity's `CRITIQUE` stage seems to have led to different kinds of discussion and consideration of diverse perspectives.

Teams perceived limitations uncovered by CIDER as meaningful to address, but recognized feasibility tradeoffs

I positioned the collaborative CIDER activities within the class as one way for teams to identify the limitations of their project prototypes, which they were required to address in a section of their final design specification writeups. Some teams found the CIDER activities very useful in identifying critical limitations, assumptions about users, and biases that they had previously overlooked:

> Team I: *" [The CIDER activity] highlighted limitations in regards to accessibility and individual user abilities. It helped us find things that we didn't think of initially and going back and looking over our prototype helped us come up with a more inclusive version of our initial design."*

Some teams were more critical of the activity's usefulness, reporting that CIDER really only helped them find minor limitations:

> Team C: *" [...] we didn't find them [the limitations] to be super significant or that would affect the entire design. They were minor changes that might help us remove some assumptions and make our design more inclusive."*

When asked whether they felt these uncovered limitations were actionable, teams recognized the tradeoffs and tensions inherent in trying to improve inclusiveness when certain design decisions had already been made. For instance, Team M's project involved a smartphone app which required Internet access to function:

> Team M: *“[T]here would definitely be unavoidable tradeoffs. [...] While we could hopefully design around some of these problems and make it more accessible, basing our design online makes it inevitably hard for us to address some basic issues. Things like access to the internet, smart devices, electricity, the need to be able to read and function are all issues that ultimately can’t be fixed with our current project.”*

A handful of teams noted the particular constraints of analytical design evaluation methods like CIDER: It was difficult for them to really know how impactful the limitations they had uncovered might be without the input of actual users. One team suggested that the assumptions they identified with CIDER might even guide targeted recruitment for future empirical design evaluations and user studies:

> Team L: *“ [I]f we had more time we would definitely try to get more diverse individuals (based off of our list of assumptions) to interact with the prototype.”*

Critiquing teams’ own artifacts was difficult, but prior CIDER experiences made it easier

When asked to describe how their experiences differed when using CIDER on their own designs, almost all teams reported that they found it more difficult to critique their own designs. Sometimes, teams reported that this difficulty arose from being personally invested in the design decisions they had already made, which made unbiased critique difficult. Other times, teams noted that because they were so immersed in their projects, it could be difficult take a step back and interrogate their own assumptions:

> Team I: *“We think it is easier to critique someone else’s design because when it is your own design it can be hard to find faults if you are set in your own perspective. When you spent so much time constructing the prototype, you’ve developed your own view on it and it can be hard to break free from that.”*

This is somewhat similar to the *expert blind spot* phenomenon [214] which arises when expertise and experience prevent a person from perceiving difficulties which early designers might encounter. Finally, some teams noted that they had already tried to address the assumptions they could think of during their initial design process, which made it challenging to identify even more assumptions that they had not yet considered during their collaborative CIDER activity:

> Team A: *"I feel like with our own design, we already tried to address a lot of the assumptions when designing it. If we didn't already find assumption problems in the process of designing the prototype, it's harder to spot them out in this stage."*

In order to surpass these difficulties, teams often relied on experience they had gained from previous CIDER activities. Several teams noted the usefulness of having conducted prior CIDER activities in helping them identify a variety of different user attributes that they would otherwise not have considered when designing:

> Team B: *"It [the CIDER activity] helped us become more open-minded and consider more perspectives; for example, we learned to consider factors such as physical space, access to different resources, and mental capabilities of the users. Lastly, it helped us think about physical abilities that other users might not have that we take for granted."*

Other teams leveraged outputs of the previous CIDER activities to help surface more kinds of potential bias. For instance, Team E returned to the assumption lists they had created during the `CRITIQUE` stages of previous CIDER activities for inspiration:

> Team E: *"At the start of brainstorming our group really felt we had a practically perfect design. We had to go back to what we had written about other designs [on previous CIDERs] for ideas. In reviewing those answers we realized how our design was also challenged."*

Multiple teams also mentioned that the collectively-generated assumption lists from the individual CIDER activities served as a useful resource, especially when it came to identifying "common" assumptions that arose in multiple prior activities:

> Team K: *“ Having practiced critiquing other design’s has made us more critical and more aware of common design assumptions. Further, reading other people’s responses to the [CIDER] Inclusive Design Activities were extremely beneficial in providing multiple perspectives and types of assumptions that we had previously not considered.”*

When teams used the outputs of prior CIDER activities to help them better uncover assumptions in their own designs, they reported being better able to recognize potential design bias. Notably, the teams who used these strategies did so without any particular prompting from the instructor to reflect on prior CIDER results. Taken together, these results suggest that not only can the CIDER technique be useful in equipping students with skills to recognize design bias, but also that the outputs of CIDER-based activities might be beneficial in making future design processes more inclusive as long as they are readily accessible.

Prior CIDER experience helped teams make their designs more inclusive from the outset

Several teams mentioned in their reflections that their previous experiences with the individual CIDER activities helped them become aware of more types of assumptions:

> Team N: *“All of our inclusive design activities [prior individual CIDERs] contributed a lot to how we approached our design assumptions this week [team CIDER]. From all of our practice with the past design activities, we were able to approach the assumptions from many perspectives: user goals/needs, a user’s prior knowledge, and assumptions about their abilities. [...]”*

As a result, when they were working on their own design projects, many teams strove to avoid making these kinds of assumptions about users in the first place, recognizing the assumptions early on in their design processes (e.g. during low-fidelity prototyping) and addressing them by changing their designs to be more inclusive.

Some teams pointed out that even though they had benefited from the individual CIDER activities early on in their design processes, the team activity was even more helpful, because it gave them the prompting and scaffolding necessary to identify more nuanced kinds of assumptions that they might have otherwise overlooked. For instance, Team O identified a narrower range of assumption types in their collaborative CIDER activity's `CRITIQUE` stage than many other teams. However, they attributed this not to a lack of ability to recognize bias, but due to the fact that they had already identified and addressed some of their unconscious assumptions about users early on in their design process:

> Team O: *"[W]e tried to identify these assumptions early on in this project and tackle as many as we could through our original design. Previous experience has helped us think of these assumptions in a wider scope, through not only target user feelings but also others who engage with this design. [... This week,] We tried to generate assumptions that could be tackled with a more solid solution that works with our prototype."*

Taken together, these results suggest that the CIDER technique may help students design more inclusively because it alerts them to common inclusion pitfalls. In a collaborative setting, CIDER-based activities might even provide teams a mechanism to talk about more often overlooked kinds of inclusion, even when teams consider their assumptions from the very beginning of the design process like Team O did. This especially promising given that having more experience with CIDER seems to contribute to more inclusive design processes overall, indicating that regular CIDER-based activities might build up early designers' knowledge base of design bias examples over time, and help them be better equipped to either avoid it in the first place, or respond to it when it does manifest.

5.5.4 RQ4: What kinds of lasting impacts might the CIDER technique have on students' design approaches?

To understand the ways in which CIDER activities might have had longer-term or broader impacts on students' understandings of inclusive design, I conducted post-class interviews about a month after the term concluded. These interviews asked students to reflect on how their own attitudes toward design and inclusion changed over the course of the term, as well as how the activities might have influenced their personal approach to design work or to other aspects of their lives.

Students gained new understandings of design and the importance of inclusion

Many students mentioned gaining a growing appreciation for design in general, which helped them check their stereotypes about the discipline. Prior work indicates that computing students often hold misconceptions about the discipline of design, such as that it is only about aesthetics or that it lacks rigor [229]. I saw this reflected in the interviews. Some students were initially intimidated by design due to these misconceptions, though they later grew to understand and appreciate the complexity of design:

> P22, interview: *"[Originally] I felt like I was not born for this. [... like] I can't visualize where stuff should be and how it should actually look for an actual product. [...] I first thought that design was more about the look, the aesthetics. Well, after taking the class, I also know that the user experience is so important and that a navigable interface is more of what design kind of is. Rather than just the aesthetic itself."*

Other students recognized their prior prejudices against design and were able to reframe their conceptions of the discipline, especially when they realized just how much design work goes into creating a technical artifact:

> P29, interview: *"While I knew that design was important and complex, I wasn't aware of the depth of the issues that design entails. This is mostly due to my lack of design experience before this class, but I also think that design is often looked down upon by the rest of the STEM community as being too "artsy" and feminine and while I had tried to be aware of those biases I still think I was mildly prejudiced against design due to that. [...] My perspective on design in general definitely improved. By seeing the specific ways design is used and its importance, I was able to let go of a lot of the biases I had associated with design and respect it a lot more."*

Not only did students gain better understandings of design overall, but many of them specifically reported better understandings of design bias, design exclusion, and the importance of inclusive design. Some students who were less familiar with inclusive design at the start of the class relayed how they broadened their definition of inclusion over time to more than just conventional accessibility concerns:

> P40, interview: *"I ha[d] heard of the term inclusive design before but not familiar with it. I considered it as design that aims to make more people accessible to it and specifically it could benefit disabled people. [...] I got to know what inclusive design really represents and how it works to increase the accessibility at the same time eliminating exclusions from users, which really attracted me."*

Others mentioned their surprise at the complexity of practicing truly inclusive design when creating an artifact, even if they were ideologically committed to access and inclusion in the first place.

> P29, interview: *"I already was well aware of the importance of accessibility and inclusivity in technology (and just in general). I think the only thing that really changed was that I gained a better understanding of just what goes into achieving those kinds of technologies and how complicated and difficult it can be, especially when a lot of the technology field doesn't realize its importance."*

CIDER activities challenged students to consider new perspectives on inclusion

When asked what aspects of the course had had the most impact on students' newfound understandings of inclusion, some students specifically attributed their newfound appreciations of inclusive design to the CIDER activities[6], even before they were prompted to consider the CIDER activities in particular.

[6]Within the context of the course, CIDER activities were simply referred to as "inclusive design activities." As a result, quotes from the interviews refer to inclusive design activities rather than CIDER activities. These phrases should be interpreted as interchangeable in the context of students' quotes.

> P12, interview: *"I didn't have much experience with inclusive design before the class, but I feel as though I have learned and understood the importance of inclusive design over the course of the class. [...] The weekly inclusive design activities were very helpful to put things into context and put skills into practice."*

When the interviewer prompted students to reflect upon the CIDER activities in particular, many students shared positive feedback and highlighted the activities' roles in changing their perspectives on inclusive design:

> P3, interview: *"I LOVED these activities. I think that it gave us students first-hand experience to show us exactly how difficult and important inclusive design is without making us feel overwhelmed by the assignment. I learned how to better identify assumptions that designs were making and how to think of more solutions to address an assumption. [...]"*

> P12, interview: *"I enjoyed the inclusive design activities a lot and I feel like I learned the most from these activities. They helped me learn about the assumptions that common products make about their users as well as assumptions that I could potentially make. I would not have thought of and reflected on these assumptions as much if these activities were not a part of the class."*

When asked if any particular parts of the CIDER activity especially contributed to their learning, two students replied that all of its parts were helpful, as P29 did:

> P29, interview: *"The listing out of assumptions forced me to challenge my understanding and really try to notice things that I wouldn't have noticed at a glance. The scenarios helped me to connect the issues to real life and understand the negative ways the flaw could impact people and which people would be impacted by it. Brainstorming redesigns led to a better understanding of just how difficult it is to account for people's needs and just how ableist design standards are. Seeing other people's*

> *designs helped me to understand what things I tended to miss and try to change my perspective so that I would be more aware of those things."*

Other students specifically emphasized the benefits of the CIDER activities' *EXPAND* stage, where the educator collates and shares back the collectively generated list of assumptions that students all came up with. This stage allowed students insight into new perspectives and built out their knowledge bases with rich examples of design bias and exclusion, which they could then apply in future design processes.

> P12, interview: *"Listing out the assumptions and seeing others' assumptions were very useful to me. It was interesting to see what I could come up with regarding what a product assumes about the user, because this also reflected what I assume about the user in a sense. It was also interesting to see what other classmates had come up with because there was always at least one thing I had missed."*

> P40, interview: *"I felt seeing how others could think of so many ways to make a design more inclusive gives me 'aha' moment, especially when I was limited of coming up ideas."*

CIDER helped students recognize their own biases and adjust their approaches to design and to life

Finally, the interviewer asked students to share their biggest takeaways from the course, as well as whether anything they had learned had influenced their current approach to design. Some reiterated their changed perspectives on design as a discipline, and underscored their reframed perspectives of its importance, especially around accessibility:

> P29, interview: *"My biggest takeaways from [course] were that I had prejudices against design that I didn't even realize, that I actively needed to change those biases which is exactly what this class did for me, and that while I had tried to educate*

> *myself on accessibility issues I still had holes in my understanding which this class helped fill."*

Others shared takeaways directly related to the importance of designing for users with differing capabilities and contexts, sometimes tying these realizations directly to the CIDER activities:

> P12, interview: *"The [CIDER] activities definitely played a role in changing my perceptions of design in that they made me realize that as a designer, I can't make hasty assumptions about my user and I should strive for my designs to be inclusive of as many populations as I can include."*

> P3, interview: *"Inclusive design when done successfully is extremely powerful and can really help improve the lives of those that the design is trying to serve. Design is directly linked with empathy. Inclusive design is no longer a choice. It is something that every design needs to consider and do."*

In terms of longer-lasting impacts, two students reported that they had applied their newly-gained knowledge of inclusive design to design projects they had worked on since the class's conclusion. For instance, P28 shared how they had used aspects of what they had learned through the CIDER activities to adapt a personal project they were working on:

> P28, interview: *"The assumptions other students had listed helped me realize and think about constraints other people might have and see which I would not have noticed myself. I used some of those in making my [current, personal project's] design more inclusive."*

Finally, some students reported that the course and the CIDER activities had influenced their daily lives beyond just their design work, such as in their awareness of design exclusion or their consumer habits:

> P40, interview: *" I felt I was better at identifying assumptions and coming up with solutions to increase the accessibility for a design at the end of the course. Also I*

> *began to pay attention to objects beside me and think of how they could exclude certain users from using them and could be designed to become more inclusive.”*
>
> P3, interview: *“I think that I am just a lot more aware of the designs of products that I use. It has also shifted my purchasing habits – I wish to support companies that put in genuine efforts in inclusive design. I also evaluate the design of products that I use sometimes which was something I rarely used to do.”*

5.6 Discussion

My investigation of the CIDER assumption elicitation technique explored how this method might help early designers learn inclusive design skills. The novelty in the CIDER technique’s approach is twofold. First, it uses the lens of *assumptions about users* as a focus for students’ critique, which is uncommon in design evaluation methods, and especially rare amongst educational methods for early designers. Second, the technique attempts to support students in *reifying*, or making concrete, understandings of how abstract ideas like inclusion are tied to actual design decisions, enabling the development of actionable inclusive design skills that can generalize across different contexts.

5.6.1 Summary of Key Results

RQ1: Students in the case study grew more and more confident in their abilities to practice both general and inclusive technology design over time. Nonparametric statistical analyses conducted on students’ self-reported self-efficacy scores for 13 design skills showed significant increases on all skills from Week 1 of the course compared to Week 10. Many students reported that the CIDER activities played a role in building their confidence. For some students with very low initial-self efficacy, the series of CIDER activities provided a feedback mechanism for them to concretely observe their own progress over time (identifying more types of assumptions, brainstorming more redesign proposals, etc.). For others, the CIDER activities may have initially *decreased* confidence in their ability to design inclusively, because the assignments revealed just how nuanced and difficult inclusive design could be. However, after subsequent practice with later CIDER activities, these students reported recovering their confidence and gaining actionable understandings of inclusive design. The impacts of CIDER activities on students' self-efficacies are impossible to isolate due to the way they were integrated within the course, but students' self-reported perceptions indicate some potential for CIDER to help build inclusive design confidence.

RQ2: After completing a series of activities based on the CIDER technique, students in the case study were able to recognize and respond to many different types of design bias in software and hardware interfaces. Students surfaced embedded assumptions in a variety of HCI artifacts related to users' potential prior knowledge, their physical and mental capabilities, and their surrounding contexts, resources, and environments. Within the scope of any given CIDER activity, the particular nature of the artifact under critique seemed to influence the kinds of assumptions students surfaced, likely due to salience of different interaction styles in the artifact's design. Over time, students demonstrated increasingly more comprehensive understandings of design bias types, with the largest increases happening after the first individual CIDER activity. The collectively-generated lists created in each activity's `EXPAND` stage, which provided a means for students to engage with peers' assumptions, seemed particularly instrumental in broadening understandings of the assumption space.

RQ3: When I tested out a collaborative version of a CIDER activity, where student teams critiqued their own project prototypes, teams often initially struggled to objectively evaluate their own artifacts. To circumvent these difficulties, many teams relied on their prior CIDER activity experience, recalling assumptions they had identified before and returning to the existing `EXPAND` lists from previous individual activities for examples of different kinds of assumptions. Several teams felt it easier to surface a wide range of embedded assumption types with their teams due to the diversity of knowledge and experience that teammates contributed to the discussion. When reflecting on their experiences with the collaborative CIDER activities, teams demonstrated nuanced recognitions of design tradeoffs around the feasibility and ethics of inclusive design. Some teams relayed that their experiences identifying and responding to design bias from previous individual CIDER activities helped their group make their design more inclusive from the outset, enabling them to avoid common pitfalls that might present accessibility barriers for users. For these teams, the collaborative CIDER activities provided a mechanism for teams to more deeply reflect on their own unconscious biases and identify assumptions about users' broader contexts that they may not have had the opportunity to discuss otherwise.

RQ4: In post-class interviews, students shared that the insights they had gained through the CIDER activities had lasting positive impacts on their perceptions of inclusive design. Some students shared that the CIDER activities had prompted them to consider perspectives on inclusion that they would not have thought of otherwise, leading them to challenge and change their own conceptions of design work. Students who continued practicing design after the course (through design internships or work on personal design projects) reported that they had adopted more inclusive approaches to their current design processes as well. The CIDER activities even influenced some students' everyday lives, with one student reporting more attention given to the designs of objects surrounding them and another describing a shift in consumer habits to support companies that valued inclusion. Overall, the use of the CIDER technique seems to have had long-lasting impacts on students' approaches to design work and everyday life that extended beyond the classroom context.

5.6.2 Limitations

Though the data I collected throughout the case study was rich, some aspects of my study design limit the generalizability of my findings. As mentioned in Section 5.4.1, this study took place during a term of remote learning necessitated by the COVID-19 pandemic. Educators and students had to adapt to teaching and learning in new, different ways at a time when external stressors were extremely high. These considerations certainly influenced the kind of data I was able to collect when evaluating my technique, and likely the engagement and kinds of responses students gave to activities and surveys as well.

Further, several limitations arise from the context of the course within which I conducted the case study. The course was taught at a university with a strong design culture which valued accessibility, by an instructor who already had relevant expertise in inclusive design and design evaluation methods. Post-secondary computing instructors who do not have expertise in accessibility-related topics or who lack institutional support may face difficulties integrating inclusion-related content into their teaching [165, 265]. The instructor also had the freedom to design a course structure that allowed for multiple repetitions of the CIDER activity throughout the academic term, which is not always possible for HCI educators due to time or curriculum constraints [302]. If a similar case study were to be conducted with a different instructor, different students, at a different institution with a different design culture, the findings might differ based on the nuances of each learning environment.

The means and methods by which I chose to explore students' experiences using the CIDER technique are inherently accompanied by their own limitations. I used digital surveys, digital assignments, and interviews conducted over video calling platforms or email to understand students' perspectives. The digital media I used to collect data may have influenced what or how much students shared. Much of my data was self-reported by participants, and thus is somewhat limited by students' individual abilities to reflect upon their own experiences (which can differ from person to person). As with all research that involves human participants, there is always a risk of response bias. This risk can be heightened in educational contexts where there is a power differential between educators and students, because students might be inclined to respond in particular ways to try and improve their grades or class standings. I tried to minimize response bias in several ways, such as by grading the CIDER activities based on participation only, by withholding survey responses about the activity from the instructor until after the course concluded, and by having a researcher unaffiliated with the course conduct the follow-up interviews.

Due to the way the CIDER activities were embedded within the course, it is impossible for us to make direct causal claims about the technique's impact on students' design knowledge. Conducting multiple individual CIDER activities prior to the collaborative CIDER activities certainly impacted students' processes on the collaborative activities (oftentimes for the better, as discussed in the RQ3 results), but choosing to structure the study in this way does not allow us to isolate the individual activities' specific impacts. I also cannot know for sure how much the ordering of artifacts within the series of individual CIDER activities impacted the development of students' inclusive design skills, or how exactly the activities interacted with the other aspects of the course to promote learning. To address these kinds of limitations, I collected data from a wide range of sources and used it to triangulate and lend credibility to my findings. However, future work remains to isolate the exact effects the use of the technique and the structure of activities might have on students' learning.

Finally, I acknowledge the research team's positionalities as researchers and educators with regards to our technique design, study design, data collection, and data analysis decisions. Each member of the research team brought their own perspectives, content knowledge, and lived experiences to bear on each part of the study design and data analysis they were involved in. My interpretations of the findings, the data I chose to collect, and even our research questions themselves may have been different if the research team comprised of different people with their own backgrounds and lived experiences. Further, our interpretations of the results (especially those from the qualitative analyses) are influenced by the individual points of view from which we engaged with the data. I also recognize that there are particular perspectives and experiences that the research team does not have access to based on our positions within academia and within society as a whole. Future work exploring inclusive design learning should also be sensitive to the ways in which researchers' and educators' standpoints might influence the questions investigated, data collected, and findings surfaced.

5.6.3 Considerations when Using CIDER to Teach Inclusive Design Skills

Choosing artifacts to highlight different kinds of design bias and exclusion

As discussed in my RQ2 results, the artifacts students critiqued influenced the types of assumptions that they identified during the `CRITIQUE` stages of the activities. Students tended to identify assumptions that corresponded to the artifact's most salient interaction styles—for instance, commonly identifying assumptions related to a user's hearing when analyzing the Google Home voice assistant, overlooking hearing-related assumptions when critiquing the QWERTY desktop keyboard. These findings suggest the potential for "targeted" CIDER activities that might help students recognize and ideate on specific facets of accessibility and inclusion.

These results also underscore the importance of selecting the artifacts used in CIDER activities carefully, with particular attention given to the way an artifact's interface might make certain types of assumptions more or less obvious to students. This consideration is especially important if educators opt to do only a single activity using the CIDER technique in their classes. For instance, if the only artifact students analyzed with CIDER was a browser-based website interface (similar to my case study's Week 4 activity), students might handily identify embedded assumptions about users' prior knowledge, visual ability, context, or language fluency; but they might not as readily consider assumptions made about a user's hearing, motor, or cognitive abilities (see Table 5.5, row 4).

Though there is likely not a single HCI artifact that can make all types of assumptions salient, a series of CIDER activities that used the technique to showcase biases in artifacts with diverse modes of interaction might be the most effective in helping students gain a comprehensive understanding of design bias. In my case study, I saw promising results from choosing five different artifacts whose major interaction paradigms and affordances varied: By the final of the five individual CIDER activities, most students had identified most types of exclusionary assumptions at least once, indicating a base level of familiarity with those particular kinds of design bias. Future applied work involving the CIDER technique might explore the creation of a set of activities which, when done in sequence, provide opportunities for students to learn about and identify a broad range of embedded assumptions. This kind of resource might prove useful to HCI educators for use in their own classrooms.

Critiquing real-world artifacts vs. critiquing students' own prototypes to surface unconscious biases

The theoretical grounding of the CIDER technique rests partially on the notion that students should critique HCI artifacts that were designed by others. As discussed in Section 5.3.1, analyzing existing HCI artifacts helps students understand the need for inclusive design by revealing the ways that real-world designs can be biased against different types of users. Practically, having all students critique the same artifact also enables the creation of the collectively-generated `EXPAND`-stage list of assumptions students draw upon to build their knowledge bases for design bias. As referenced several times by students throughout my findings, the `EXPAND` lists seemed to play an important role in students' learning.

However, in the collaborative CIDER activities (RQ3), students had the opportunity to reflect on their own biases by using the CIDER technique on their own prototypes. This kind of self-reflection provides an opportunity for growth if a student (or team) recognizes and moves to mitigate their own, likely unconsciously held biases. Indeed, though teams reported that critiquing their own prototypes felt more difficult than critiquing existing artifacts, many of them also felt that they gained deeper understandings of their own biases and how they might manifest in design exclusion. This led them to modify their prototypes to increase inclusion and accessibility. Being able to reliably provide opportunities for the kind of critical self-reflection that leads to students recognizing their unconsciously held beliefs about users would be a powerful tool in any HCI educator's toolkit—though it is unlikely that the collaborative CIDER activities would have been as successful at doing so without the previous experience students gained from the individual activities. Future adaptations to the base CIDER technique might explore a combination of activities involving real-world artifacts and activities critiquing students' own prototypes, perhaps gaining the benefits of each approach.

Conducting activities individually vs. collaboratively to surface different perspectives

The individual CIDER activities students conducted throughout the term enabled them to become more aware of assumptions in their own design work (RQ2). When it came time for the collaborative activities, some teams surfaced fewer embedded assumptions overall during CIDER's `CRITIQUE` stage because they had already anticipated and addressed these assumptions in their initial design processes. On the other hand, some teams still found it difficult to self-reflect upon their own artifacts, indicating that the scaffolding provided by previous individual CIDERs might not have been enough for students to feel confident in surfacing assumptions (RQ3).

Building upon these insights, HCI educators might consider whether individual or collaborative CIDER activities (or both) best fit their learning contexts, as well as modifications to surface new perspectives. For instance, due to the variations in teams' project designs and some inherent constraints upon the format and timeline of the course, I did not deeply explore the `EXPAND` stage of the collaborative CIDERs. To provide a source of feedback and surface more assumptions, teams could be paired up and conduct collaborative CIDER activities using each others' artifacts. One could also imagine a modified collaborative activity where students `CRITIQUE` their team's design separately, then bring their list of identified assumptions to a team meeting to discuss. Yet another modification might involve alternating between individual and collaborative CIDER activities so students could engage with others' insights through the original `EXPAND` stage list format and through peer discussions, perhaps contributing to faster-developed or more comprehensive understandings of inclusive design. Future work exploring these kinds of modifications might surface insights about teamwork in post-secondary inclusive design education as well as professional team-based design contexts.

When conducting collaborative CIDER activities, educators should also be mindful of diversity among design teams. In the study, I allowed students to self-select into their project teams according to shared topics of interest. However, as mentioned in my RQ3 results, many teams reflected upon the benefits of working with students with different levels of design knowledge, backgrounds, and lived experiences, often underscoring that the diversity of perspectives amongst the team enabled them to identify more varied types of assumptions than they might have individually. Collaborative CIDER teams specifically created with student diversity in mind might be even more effective at identifying and responding to embedded assumptions.

Teaching inclusive design skills in different learning contexts

The CIDER technique was designed to address specific teaching and learning difficulties shown to arise in formal, post-secondary computing education contexts, such as HCI courses within computer science or information science programs. This choice was due in part to the expertise of the dissertation author and analysts, but also the fact that computing departments often do not support students in developing holistic understandings of how software design impacts end users (see Section 5.2.1 for some examples). Given the continuing enrollment increases in computing programs [215], more and more students are graduating and entering careers where the technology they create might directly impact the lives of hundreds, thousands, or even millions of people. Instilling some sense of design responsibility through effective inclusive design education within these programs may help mitigate future potential harms.

On the other hand, formal computing education programs are certainly not the only sites of software interface design education. At several universities, entire courses of study focus on HCI/UX design and technological interaction design. These curricula might integrate one or more courses or modules that focus specifically on inclusive technology design. Correspondingly, the attitudes of students in these programs likely differ significantly from students in computing programs, such as in their initial understandings of how design impacts users, the role of designer subjectivity, and students' willingness to embrace inclusive design paradigms. In these cases, the CIDER technique might be adapted to better target the inclusion literacy levels of students. For instance, future work might explore whether it is viable to have HCI/UX design students analyze assumptions within their own designs from the outset during CIDER's `CRITIQUE` stage, as presumably these students already understand that inclusion issues exist in real-world technologies. Students in full UX design programs might also benefit from extensions of the method that integrate actual prototyping of proposed solutions from the `DESIGN` stage that improve inclusiveness, enabling even more concrete understandings of the nuances of inclusive design.

Promoting inclusive design agency and responsibility

As discussed in my analysis of the post-class interviews (Section 5.5.4), some students reported that the use of the CIDER technique impacted their approaches toward design work, leading them to value inclusion and attempt more inclusive practices during subsequent design projects at internships or for personal portfolios. Many of my students' reflections had a similar tone to those relayed by Kharrufa and Gray in their investigation of HCI *threshold concepts*—topics that, once understood, irrevocably change a person's perception of a discipline and their role within it [172]. These longer-lasting impacts on students' attitudes toward inclusive design are extremely promising for the efficacy of the CIDER technique in promoting a sense of responsibility for accessibility and inclusion among early designers.

However, prior work investigating early designers' development of competence at their first professional design jobs has noted the role of the surrounding organization in enabling (or preventing) early designers to enact their values in their actual design practice [125, 295]. Simply instilling a sense of responsibility for inclusion in early designers through classroom instruction may not be enough to enable them to overcome external pressures after graduation. Future work involving the CIDER technique might explore its ability to encourage students to move from simply recognizing their responsibilities as designers to becoming active advocates for accessibility and inclusion, similar to the "Activism/Advocacy" trajectory of designerly identity described by Chivukula et al. [54].

5.7 Conclusion

The CIDER (*Critique*, *Imagine*, *Design*, *Expand*, *Repeat*) assumption elicitation technique is an educational analytical design evaluation method to help early designers learn inclusive design skills. I evaluated the efficacy of this technique in a series of individual and collaborative activities in an introductory post-secondary interface design course, finding that it enabled students to recognize and respond to many different types of design bias. In follow-up interviews several weeks later, some students reported that their experiences with the CIDER technique had had long-lasting positive impacts on their approaches to design, encouraging them to value inclusion and consider more diverse types of users in their subsequent design work.

While this is only an initial evaluation, the results from the case study indicate great promise for the use of the CIDER technique in classroom contexts and potentially even beyond them. By using the lens of *assumptions about users* as a means to reveal instances of design bias and potential exclusion, techniques like these can help instill appreciations for and commitments to inclusive design early on in design education. In turn, these designers can contribute to the creation of more inclusive HCI artifacts, whether in professional practice or even just in their own personal design work. While education is not the only tool that will be necessary to build more inclusive futures for technology design, techniques like CIDER are a crucial initial step along the path to a world where everyone, not just stereotypical “average users,” can equitably and effectively interact with technology.

Chapter 6
PEDAGOGICAL CONTENT KNOWLEDGE FOUNDATIONS FOR TEACHING CRITICAL COMPUTING INTERFACE DESIGN

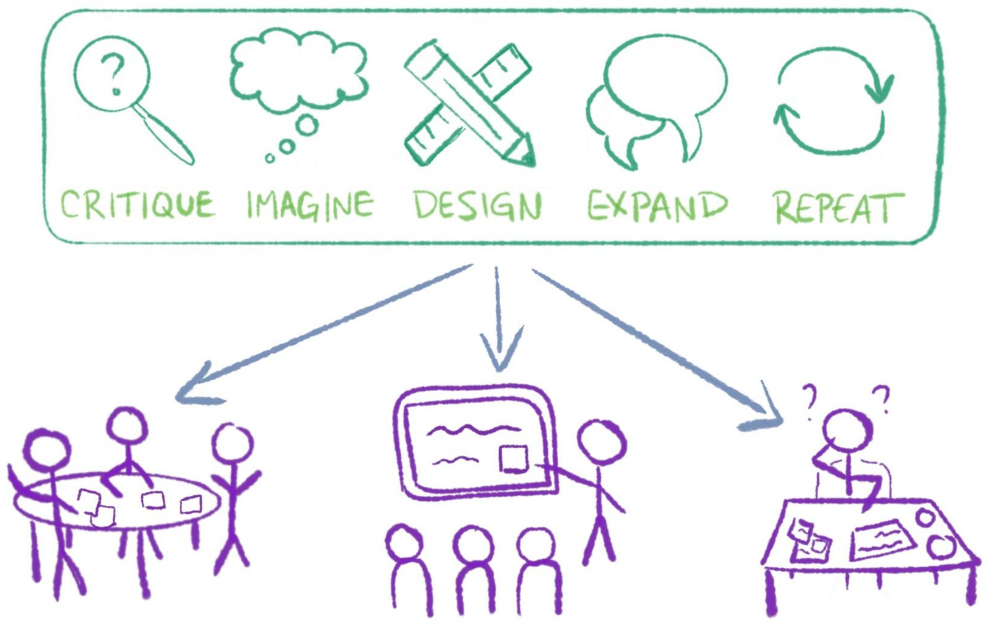


Figure 6.1: The study described in this chapter contributes an Action Research investigation of how computing educators used the CIDER technique from Chapter 5 to teach critical computing interface design in their courses.

6.1 Introduction

To realize more equitable technology futures, it is not enough to simply adapt technology to be more inclusive *after* it is created: We also need to equip technology creators with the skills they need to critically reflect upon bias and exclusion *during* the technology design process [111]. It stands to reason that formalized computing education could be one way to develop these skills in the students who will become tomorrow's technology creators and consumers [228]. Many factors influence whether computing professionals are able to enact justice-centered design values in practice [54, 295], but recent work indicates that integrating inclusive design topics into post-secondary computing education can have positive lasting impacts on students' technology design approaches [230].

Of the myriad different types of design that go into creating a technological artifact, *interface* design can be an illustrative site of inquiry for demonstrating the concrete impacts of design bias. Computing interfaces are the point at which technology designers' conceptions of the world meet the realities of users' everyday lives. Interfaces constrain allowable interactions with technology [306], and by extension, who gets to benefit (and who is harmed through exclusion) from technological access. As human-made artifacts, computing interfaces embed the values and biases of their creators [100, 101], often leading to technology designs that fail to meet the needs of users from different marginalized groups [69]. However, because technology carries perceptions of objectivity [18], it can be difficult for students to grasp how subjective design decisions made throughout the design process might result in more or less usable technology [230, 268]. Helping students learn to critically reflect upon the ways their design decisions impact users can be one aspect of developing inclusive design competence, and by extension, support critical computing education efforts [178, 313].

To teach these topics effectively, we require well-developed pedagogical understandings upon which to base resources and teaching materials. As it stands, we largely lack these foundations. A 2019 survey of computing educators indicated that many were interested in integrating topics like accessibility and inclusive design into their courses, but did not, because they did not know where to start [165]. In education research, this missing knowledge is often referred to as *pedagogical content knowledge* (PCK) [266]. PCK is domain-specific [128, 146, 150] and consists of knowledge of pedagogical strategies to teach a particular *topic*, in a particular type of *learning context*, to a particular *audience* of learners. Exact definitions of the components of PCK vary (c.f. [29, 108, 211]), but knowledge of potential instructional challenges and how to address them is generally considered a core component. Within the last decade, prior work in computing education research has begun to investigate and document different manifestations of computing PCK. However, though some recent conceptualizations of critical computing education include interrogation of the implications of technical design decisions as a key competency [195, 212], we lack in-depth PCK-focused explorations of the topic.

Toward the development of PCK foundations, I conducted an Action Research [136, 276] study with a community of computing education instructors and researchers with existing commitments to integrating critical[1] interface design topics into computing education. The community elected to use the CIDER assumption elicitation technique (see Chapter 5) as a focal pedagogical method due to its explicit focus on helping computing students learn to critically reflect upon the disproportionate impacts of design decisions on different user groups. Over the course of 22 weeks (two academic terms), I supported three computing instructors and their teaching assistants (TAs) for three different courses in integrating CIDER-based activities into their computing curricula. To scope the investigation, I elected to focus on a specific element of critical computing interface design PCK [29, 228]: the *instructional challenges* instructors encountered and the *mitigation strategies* they employed to address these challenges. This resulted in the following research questions:

1. What **instructional challenges** arise as computing educators integrate critical computing interface design topics into computing courses?

2. What **mitigation strategies** do computing educators employ to address the above challenges?

To document instructors' experiences and reflections on these topics, I conducted a series of semi-structured interviews, observed their courses, co-designed learning materials, and facilitated

[1]For this chapter, I define "critical" interface design work according to the same justice-centered conceptualization of inclusive design I describe in Chapter 2: A manner of approaching interface design that considers the implicit power dynamics and norms of interface creation and actively opposes hegemonic tendencies to design solely for the needs of dominant user groups. I consider inclusive design as one aspect of critical computing interface design work. The instructors I worked with in this study also conceptualized what they were doing as "critical" design rather than "inclusive" design, so I have elected to remain authentic to their characterizations.

a community support Slack channel. Iterative qualitative thematic analyses of data from the above sources surfaced eleven distinct types of instructional challenges, each of which had a *general instruction* component that might manifest in any computing or design classroom, and a *critical-specific* nuance that was unique to the teaching and learning of critical interface design topics within computing education. For each of these challenges, I also identified one or more mitigation strategies instructors used to address it, for a total of 23 strategies presented in this chapter. The insights from this investigation provide pedagogical content knowledge foundations for inclusive computing interface design education and contribute to broader discussions around best practices for teaching computing students to critically reflect on the impacts of technology they create. [2] [3]

6.2 Related Work

6.2.1 Existing Efforts: Teaching Critical & Inclusive Computing Interface Design Principles

Within computing education, there exist myriad approaches to teaching critically or teaching critical content. Morales-Navarro and Kafai historicized and described several prevalent approaches in their recent review, proposing that there exist three major pedagogical operationalizations of criticality in CS education: critical *inquiry*, where students interrogate and deconstruct hegemonic computing narratives; critical *design*, where students practice justice-centered creation of computing artifacts; and critical *reimagination*, where students reconstruct and reclaim past and current computing narratives and speculate on how technology might enable more just and equitable futures [212]. They also noted that many teaching approaches draw on more than one of these

[2]Throughout this chapter, I use first-person language ("I", "my", etc.) to describe this work to reflect the single-author nature of the dissertation document. However, this work was conducted in collaboration with Jayne Everson, Rotem Landesman, Dr. Amy J. Ko, and the community of computing educators and researchers that participated in the investigation. The first two paragraphs of the related work on PCK (Section 6.2.2) in this chapter were based on prior work of mine published in the 2018 ACM conference on International Computing Education Research (ICER) [228]. The rest of the chapter content is yet to be published. I acknowledge the shared contributions of all community members to the project.

[3]We would like to thank the instructors who participated in this study for sharing their perspectives and giving up their time to help advance computing education pedagogy. We would also like to thank Noelle Brown, Benjamin Xie, Ella Sarder, Casey Fiesler, Eliane Wiese, and Jean Salac for sharing their expertise on teaching ethics and criticality within computing contexts.

operationalizations, such as design justice pedagogies [70], which leverage all three. Critical content in computing courses may also overlap with lessons on "ethics" in computing. Brown and Xie et al. conducted a systematic literature on papers involving ethics published in computing education research venues through 2022, finding that though many papers called for the inclusion of ethical or critical content in computing courses, and some even provided example materials, there was a distinct lack of consensus around best practices for teaching these topics to computing students [35]. Brown and Xie et al. further speculated that effective ethical education in computing contexts should leave space for multiple conceptions of what constitutes "ethics", noting that conceptions of ethics that only shallowly engage with diversity and social justice efforts fall short of the critical problematization necessary to challenge hegemonic norms of computing education.

Computing education and interface design topics often intersect in human-computer interaction (HCI) courses. Accordingly, some HCI education research explores how best to teach critical design concepts to computing students. Though HCI education is a relatively young discipline [56, 302], there is a quickly growing focus within it on teaching computing students to critically consider the implications of the technologies they design. Fiesler et al. give an overview of the kinds of topics that are taught as "tech ethics," noting that much of this work has been published only within the past few years [96]. Some strategies for teaching the broader impacts of technology design focus on critiquing algorithmic design biases [246]. Others focus on teaching accessibility principles in standalone design courses [198], embedded in various levels of programming courses [156, 263, 264], or integrated throughout computing curricula [293]. Still others propose to help designers better understand the (sometimes conflicting) perspectives and values of various stakeholders, sometimes through techniques like Wong and Nguyen's *Timelines* value advocacy activities [309] or Cooney's notion of *micro-exposures* [66].

Several types of student learning challenges are known to arise when teaching and learning critical interface design skills in computing-centric contexts. In general, teaching design principles to computing students can be difficult due to the ways that best practices for critical design pedagogy (c.f. studio approaches [28, 206]; "correctness" on wicked problems [94, 268]) conflict with the kinds of well-defined problems students tackle under traditional computing pedagogy [229].

Educators may not feel they have the expertise [265], time [165], or organizational supports [165] they need to properly teach accessible or inclusive interface design. Computing students might also struggle to learn basic design principles, much less critical approaches to interface design. To summarize the challenges discussed extensively in Chapter 5's related work, five major categories of challenge might inhibit students' critical design learning:

1. **Motivating criticality:** Students may devalue design work or believe inclusion issues do not really exist in "real-world" interfaces, lacking motivation to learn and practice critical design. [55, 130, 193, 228]

2. **Connecting design features to implicit assumptions:** Students, especially those with little design experience, may not recognize the connection between exclusionary interface features and the designer's (often implicit and normative) assumptions about users' capabilities and contexts. [18, 70, 101, 268, 307, 309]

3. **Designing for diversity:** Students may implicitly design for users as a homogeneous population, failing to recognize and account for diversity, especially if they do not have extensive knowledge bases of user experiences to draw upon. [7, 70, 192, 219, 228, 268]

4. **Acting on critical design goals:** Even if they value it already, students may struggle to move from abstract appreciations of inclusion, access, or ethics to concrete design actions they can take to reduce or mitigate the impact of biases in interfaces. [54, 125, 228, 295]

5. **Avoiding stereotyping:** Students may struggle to understand the perspectives and experiences of users who are unlike themselves without resorting to stereotyping, resulting in exclusionary interfaces. [20, 192, 244, 268, 302]

Any of the above learning challenges can inhibit computing students' abilities to learn critical interface design topics. This study contributes rich descriptions of several contextualized instances of these and other newly identified learning challenges, augmenting them with other instructional challenges that instructors experience when teaching critical topics. Additionally, I also contribute examples of mitigation strategies that these instructors used to address these challenges, providing exemplars that can be used to further support critical interface design pedagogy. My approach to this work aligns most closely with Lin's justice-centered approach to CS education [195], which is characterized by Morales-Navarro and Kafai as "constantly comparing dominant to CS approaches to design justice approaches particularly with regards to the values embedded in data structures and algorithms" [212]. However, in addition to describing value-based instruction in a data structures and algorithms course, I also provide insights into similar kinds of value-based, justice-centered instruction in other computing education contexts: a social media and ethics course, and a CS assessment course.

6.2.2 Pedagogical Content Knowledge: What Educators Need to Know to Teach a Topic Well

Pedagogical content knowledge (PCK) is a form of domain-specific knowledge that educators develop as they teach particular topics. Originally introduced by Shulman [266], PCK sits in the intersection of pedagogical knowledge (knowing general strategies for teaching) and content knowledge (knowing about a particular topic, such as interface design). Shulman examined teachers' experiences through the lens of what they knew and what they didn't, and how that contributed to their successes or failures in effective instruction. When expert teachers possessed some knowledge about how to teach the specific course content that novices did not, and when the novice teacher was less successful in their teaching efforts, Shulman called the knowledge PCK: knowledge of strategies to teach a particular topic, in a particular context, to a particular audience.

Myriad prior work indicates that PCK is domain-specific [128, 146, 150]. Past PCK-focused investigations in a range of fields (including literature and geography [266], chemical and biological sciences [128, 150], and math [138, 266]) suggest that even the measurement of PCK is domain-specific [137]. Recent surveys of research on PCK in STEM fields also found that teachers with better-developed PCK for their topic often saw evidence of better learning in their students [67]. Developing PCK is key to being an effective educator even when teachers have exceptionally high content expertise: In Fernandez-Balboa and Stiehl's study of PCK in higher education, they found that even the most exceptional teachers benefited from PCK despite their high mastery level of the content [95]. Exact definitions of the components of PCK vary (c.f. [29, 108, 211]), but knowledge of instructional challenges and strategies to mitigate them is generally considered a core aspect.

The computing education research field has only begun to investigate the nature of computing PCK within the past decade. Some prior work has explored PCK in primary and secondary learning environments [27, 112, 207, 243, 292]. Other computing PCK investigations have focused on post-secondary learning contexts, with some attempting to investigate the phenomenon at the level of general computing PCK [145, 146] and others scoping their studies to specific computing topics like on topics such as introductory programming [163, 169, 312], algorithms [225], or concurrency [185]. However, few of these investigations focus on teaching or learning interface design topics, much less critical or inclusive interface design.

Similarly, there is relatively scant prior work on PCK development in HCI education research, though interest in the topic is growing [302]. Recent work by Oleson et. al noted the dearth of PCK for teaching computing interface design and other HCI topics [227]. In response, Wiese et al. proposed a lightweight method for developing HCI pedagogical content knowledge in-situ through reflective conversations among instructors [300], contributing some initial generalizable insights that these conversations had surfaced. Sacré and Lallemand recently investigated HCI educators' self-reported perceptions of instructional quality as it related to technological pedagogical content knowledge (a.k.a. TPACK, an extension of the PCK model more commonly used in primary and secondary education [292]), additionally noting a distinct lack of resources and research to help HCI educators improve their teaching.

Though this prior HCI education work provides starting points for understanding and investigating teacher knowledge about computing interface design instruction, several open questions remain about best practices for general HCI instruction, and we know even less about how to teach *critical* or *inclusive* interface design principles. Sin et al. recently contributed several cases describing how their Digital Design Marginalization (DDM) framework can be used by design educators as a reflective tool for identifying inclusivity gaps in their pedagogy. They illustrated how these gaps can inhibit students' abilities to think critically about the impacts of their design decisions in their future careers, calling for more research into reflective methods to help design educators improve their pedagogy and more sharing of teaching materials that help students develop inclusive design knowledge [271]. Prior work led by Oleson et. al contributed an exploration into PCK for teaching a specific gender-inclusive interface inspection method called GenderMag [228]. However, though this investigation provided one model for HCI PCK research, its focus was broad rather than deep: It conducted several short interviews with several faculty at several different institutions across the world. In contrast, this study described in this chapter investigates the broader topic of how to integrate critical interface design topics into computing education through in-depth multi-week collaborations with a smaller, more focused community of educators.

6.3 Method

6.3.1 Action Research

This investigation into critical computing interface design PCK mirrors prior work on PCK development [228] by electing to use Action Research as the guiding methodology. Action Research is a form of longitudinal, participatory field study conducted by a community group attempting to address a specific issue important to that community. Field work in Action Research involves continuous reflection on the nature of that problem while also trying to address it [136, 276]. Action Research is unlike other kinds of empirical studies in that it does not attempt to "control" the setting being observed; instead, the goal is to attempt an intervention and learn through the outcomes of that intervention. Participants under this paradigm often act as researchers themselves, using data to refine theories of the problem, which in turn informs interventions and further data gathering. Action Research has been used in education research for decades [319].

6.3.2 Study Context, Community, and Course Integrations

In this study, the community was computing educators and researchers with existing commitments to teaching critical design content who were interested in identifying the most effective ways of doing so. Under the Action Research paradigm, the instructors I worked with acted not as participants in the traditional sense, but instead as *teacher-researchers* who helped shape the directions of inquiry over time. The instructors and researchers were all in active community with each other throughout the 22 weeks of the study. While I only officially recorded conversations in contexts where informed consent was given (like the semi-structured interviews), instructors and researchers in the community often informally chatted about new insights they were gaining through instruction and brainstormed about new ways to address unexpected instructional challenges. These informal conversations helped build out understandings of the problem, and, in turn, helped focus and refine the investigation.

The study itself took place at a large, public, North American university in the Pacific Northwest. This university is known for its vibrant, quickly growing community around computing education and computing education research. Many educators and researchers at the institution have established interests in teaching computing for social good and integrating justice- and equity-centered content into their courses.

The three instructors who I worked with in this study had been part of that educational community for several years (a minimum of three apiece), with even more experience teaching at other institutions before that. The instructors had each heard about the CIDER assumption elicitation technique described in Chapter 5 through community proximity to the dissertation author and were interested in exploring its use in their courses. When they independently expressed interest in exploring best practices for teaching critical interface design topics, I saw it as an opportunity to enter into community-based exploration of the issue.

Though many community members indirectly participated in this study through informal conversations about critical and inclusive computing education, the community members who directly participated in this study were:

- Three computing *instructors* (Instructors A, B, and C), who joined the study with an existing commitment to integrating critical content into their computing courses, though not necessarily expertise in teaching computing interface design methods.

- Three *teaching assistants (TAs)*, who either led (A-TA1, A-TA2) or helped the instructor

Instructor A: 5-10 years of teaching experience	
Viewed their role as instructor as...	Helping students build CS identities by making connections to their own experiences and learning to value peers' perspectives
Motivated to try CIDER because...	Wanted to redesign the peer feedback activities present within the course; Saw CIDER as an opportunity to try a new form of activity that showed students the limitations of their own perspectives and enabled them to learn from peers
Desired takeaways from activities	Agency to create more inclusive technologies; Bridging of the perceived gap between technical work and consideration of how tech affects society
Instructor B: 5-10 years of teaching experience	
Viewed their role as instructor as...	Equipping students with actionable skills to be informed consumers and ethically-minded future creators of social media technologies
Motivated to try CIDER because...	Wanted more design activities in curriculum; Saw potential in CIDER for students to practice speculating about more ethical technological design futures and gain understandings of tech's real-world impacts
Desired takeaways from activities	Ability to think broadly about many different kinds of people who might use a technology; Understanding the real-world impacts of interface design decisions
Instructor C: 10-20 years of teaching experience	
Viewed their role as instructor as...	Helping students recognize and dismantle implicit power structures present in traditional educational assessments, while modeling fair and equitable practices in the course itself
Motivated to try CIDER because...	Used CIDER in a previous offering of the course; Saw utility in CIDER's structured nature for helping students develop habits of reflecting upon the implicit values embedded in the assessment items they created
Desired takeaways from activities	A developing habit of critically reflecting upon who is privileged by an item's design and who is disadvantaged, to create more inclusive assessment items

Table 6.1: Details about how each of the three educators involved in the study approached their course, including motivations and goals for integrating critical computing interface design content.

facilitate (B-TA) CIDER activities.

- Three computing education *researchers* (R1 (the dissertation author), R2, and R3), who supported instructors and TAs by providing content knowledge about teaching inclusive and critical interface design methods, as well as capturing data, analyzing it for themes, and reporting back to the community on insights uncovered.

Table 6.1 gives additional details about the three instructors' pedagogical approaches, including summaries of how they perceived their role as instructor, why they were interested in CIDER, and what they hoped students would take away from the CIDER activities. I did not have a chance to interview the TAs until later in the study (after they had already taught their CIDER activities), so I did not collect the same data about goals and motivations from them, opting instead to focus our conversations on their instructional experiences. All three instructors and their TAs had previous experience teaching inclusive design topics, though only Instructor C had used the CIDER technique in a previous offering of the course prior to the study. One person involved the study acted both as an instructor and as a researcher, so the total number of Action Research community members involved in this study was eight.

Table 6.2 gives additional details about the instructors' courses, and Table 6.3 gives more information about how the instructors integrated CIDER activities (Course A = 3 activities; Course B = 4; Course C = 1). Though I did not intentionally sample for diversity in conceptualizations of computing interfaces, I did note that each instructor seemed to focus on different kinds of interfaces:

- Instructor B seemed to think of "computing interfaces" in similar ways to the evaluation in Chater 5, focusing on the user interfaces present on various social media sites and how they constrained or enabled different interactions. The artifacts in their CIDER activities were often chosen to exemplify certain assumptions involving ethical or technical lesson themes.

- Instructor A conceptualized of "computing interfaces" in a way more aligned with the software engineering usage of the phrase: The methods and functions that enable interaction

Course ID	Topics	Department	Enrollment	Course status	Typical students
A	Data structures; algorithms	CS	200+	Required for some concentrations	Undergraduates (all levels), mostly technical majors
B	Ethics; social media programming	Informatics	100-200	Elective for all	Undergraduates (1st & 2nd year), many majors
C	Formally assessing CS knowledge	Education	<25	Required for some concentrations	Graduates, all education majors & CS teachers

Table 6.2: The courses involved in this study. Course IDs correspond to the Instructor IDs listed in Table 6.1.

with a program's data, including the data structures and algorithms by which those methods were implemented. (This was confirmed in a later interview.) The artifacts in their CIDER activities included both technical (code-level) implementations of certain requirements, as well as the graphical user interface features that enabled user interaction with those implementations.

- Instructor C's conceptions of "computing interfaces" extended to the domain of education. They viewed assessments (in the sense of tests, quizzes, etc.) within CS courses as designed artifacts, with teachers as analogous to technology designers and students as analogous to technology users. As a result, the items (questions) teachers created for assessments were just as prone to bias and exclusion as any other designed artifact; and therefore ripe for critique with CIDER. The artifact in their CIDER activity was an item from the AP CS A [62] test which assessed students' knowledge of loops.

Summarizing the above differences, we can consider Instructor B's course to be a *near transfer* learning context to the original CIDER technique evaluation study [230], Instructor A's course

Instructor	Lesson topic that included CIDER activity	Artifact critiqued during CIDER activity
Course A: Course A's curriculum involved incremental development of a toy mapping and navigation application (similar to Google Maps) to teach about different abstract data structures. This maps app was the basis for two separate CIDER activities: One early on in the course, and one in the final portfolio. The two TAs co-led a special topics lesson on misinformation which also included a CIDER activity.		
A	Identify different biases in existing implementation	Maps application: Any interface feature
A	Final portfolio: Describe a more inclusive implementation	Maps application: Implemented data structures
A-TA1, A-TA2	Spread of dis/misinformation	Students' choice: Social media interface
Course B: Course B's curriculum was arranged according to certain ethically-relevant topics pertaining to social media technologies. Instructor B chose artifacts for the four CIDER activities they conducted according to what they believed would best illustrate the nuances of those themes. Course B's TA helped to facilitate the in-class activities.		
B	Authenticity	Facebook's "real name" policy (**cite)
B	Security & Privacy	Overview of GDPR data rights
B	Accessibility	Students' choice: Social media interface on own device
B	Mental Health	Students' choice: Social media interface on own device
B-TA	(Same as above, helping facilitate in-class activities)	
Course C: Course C's curriculum taught about different "lenses" that students could use to view assessment item design through (e.g. visually impaired learners, English-second-language (ESL) learners, etc) toward the goal of creating inclusive test items. Instructor C integrated CIDER toward the end of the course as a unifying framework for helping students examine test items through several lenses.		
C	Creating equitable assessments	AP CS test item which assessed knowledge of loop structures

Table 6.3: Details on how CIDER activities were integrated into the three courses by instructor.

to be a *medium transfer* learning context, and Instructor C's course to be a *far transfer* learning context. This diversity supports triangulation, which is key to establishing credibility and validity of interpretations under Action Research paradigms [276]. If similar insights arise across such varied learning contexts, we can reasonably conclude that they are at least somewhat related to the phenomenon of critical computing interface design teaching and learning, the constant across contexts.

6.3.3 Data Collected

Similar to prior work on explorations of critical and inclusive interface design teaching in computing education [228, 300], I took a pedagogical content knowledge (PCK) development lens to this investigation. Before the study began, I sent each instructor a survey to collect their contact information, basic information about their course and their prior teaching experience, and any questions they had for me (which would be addressed in their first interview). Table 6.4 shows an overview of the additional data I collected during the study, including the following:

Semi-structured interview transcripts & notes. This was my main data source for understanding instructional challenges and mitigation strategies. I conducted between one and five semi-structured 30-60 minute interviews with each instructor, in-person or over the Zoom video calling platform. After informed consent was obtained, each interview was audio recorded and (optionally) video recorded for later transcription. I led each interview and took handwritten notes to augment the transcripts, occasionally joined by the second or third researcher depending on their availability. In this manner, I collected just over nine hours of interview content. The interviews were of the following types:

- *Pre-Teach (Instructors A, B, C)*, where we discussed the instructors's approach to teaching, their prior experiences teaching critical topics in computing courses, and their goals and plans for CIDER integration. I also asked instructors about any instructional challenges they anticipated and offered space for them to ask any questions they had about the CIDER technique or about inclusive design in general, acting as sources for content knowledge around

Instructor	Semi-structured interviews	CIDER activity materials	Course observation notes	Other teaching materials	Community Slack messages	TA support session notes
A	1: Pre-Teach 2: Planning 3: Planning 4: Debrief 5: Post-Teach	2 activities	1 session	1 set	✓	1 session
A-TA1, A-TA2	1: Combined: Debrief + Post-Teach	1 activity	1 session	1 set	n/a	n/a
B	1: Combined Pre-Teach + Planning 2: Debrief 3: Post-Teach	4 activities	n/a	n/a	✓	n/a
B-TA	1: Combined: Debrief + Post-Teach	(same as B)	n/a	n/a	n/a	n/a
C	1: Pre-Teach 2: Planning 4: Debrief 5: Post-Teach	1 activity	1 session	1 set	✓	n/a

Table 6.4: The different kinds of data we collected from each instructor and/or TA, including interviews, observation notes, and teaching materials.

these topics. The design of these interviews was informed by prior work on PCK development for inclusive interface design [228].

- *Planning (Instructors A, B, C)*, which happened when instructors were ready to start thinking about the details of the specific CIDER activities they wanted to run. In these sessions, the instructor and the researchers brainstormed and co-developed teaching materials and discussed how to address anticipated instructional challenges. Instructors A and C had separate planning interview sessions, while Instructor B worked on planning activities with the researchers during their pre-teaching interview.

- *Debriefing (All instructors & TAs)*, which happened shortly after they conducted a CIDER activity in their course. I asked instructors to reflect upon what went well and what didn't, with a particular focus on any instructional challenges they experienced and how they addressed those challenges (if at all). All instructors participated in a debriefing interview, including TAs. The instructors who did multiple CIDER activities in their courses (A & B) were offered subsequent planning and debriefing interviews after this, but both declined due to time constraints and already feeling comfortable enough to continue on their own.

- *Post-Teach (All instructors & TAs)*, where I asked instructors to reflect upon the course as a whole and how they integrated CIDER within it. I asked them about what went well, what didn't, and what they might do differently if they were to run CIDER activities in the future, with a particular focus on any knowledge gaps that surfaced while teaching and any student reactions to the content. I also asked what advice they would give to a peer or colleague interested in trying CIDER activities in their courses. The design of these interviews was informed by prior work on lightweight methods for surfacing HCI PCK [300].

Co-created CIDER activity materials. In *Planning* interviews, I worked with instructors to brainstorm and co-create learning materials for CIDER activities. Most often, instructors entered these interviews with an idea of the topic or learning outcomes they wanted to target, then enlisted the researchers' assistance in deciding upon the format and structure of the activity itself. These activities served as concrete materials that revealed the outcomes of instructors' pedagogical decisions, reflective artifacts to reference in future conversations, and exemplars to share with other instructors. In total, I collected eight different CIDER activities.

Course observation notes. In two of the three courses (A and C),researchers were able to observe lessons on the days that CIDER activities took place. While observing classes, I noted salient instructional interactions around how instructors introduced, facilitated, and closed out the CIDER activities. I also noted any student questions and confusions that arose around critical interface design content, as well as how the instructor responded to them. These observation notes provided rich contextual data for understanding the instructors' pedagogy and additionally served as objects of reference during *Debriefing* interviews. R2 observed the three activities in Course A, and R1 (the dissertation author) observed the activity in Course C.

Other teaching materials used alongside CIDER activities. Instructors sometimes had additional lecture slides that introduced critical computing interface design ideas or small design activities that primed students for the more involved critiques of CIDER. I requested and obtained copies of these materials from Instructor A, Course A's TAs, and Instructor C.

Messages from the community Slack channel. To facilitate community, I invited instructors and researchers to join a shared Slack channel dedicated to the project. This provided a less-formal space to discuss ideas, challenges, and logistics around teaching critical computing interface design. Instructors used this channel to ask questions about instruction, and the researchers used it to give updates on project progress and share out newly discovered insights. In total, I collected messages from seven meaningful conversations on topics related to critical computing interface design, some of which were between instructors and researchers members, and some of which were between instructors.

TA support session notes. Instructor A, who had a team of 20+ TAs helping them teach the course, asked the researchers to join one of their TA meetings to introduce the CIDER technique. Two researchers (R1 and R2) attended this meeting, described the technique's origins, and demonstrated how a basic CIDER activity might work. Instructor A also gave an overview of how they intended to integrate the technique into the course, then opened up the rest of the time for TAs to ask any questions they had. The two researchers debriefed directly after the meeting and recorded notes on what the TAs had been interested in as well as the most salient questions that had been asked.

6.3.4 Qualitative Analysis: Affinity Diagramming and Inductive Coding

Three researchers from the Action Research community participated in qualitative data analysis:

- Researcher 1, a computing and HCI education researcher with expertise in the integration of inclusive and critical interface design topics into computing courses, as well as expertise in leading Action Research investigations and in qualitative methods. They approached this work from the perspective that enabling computing students to critically reflect upon the design decisions they make and gain inclusive design agency can help contribute to more equitable futures where everyone can access the benefits technology provides. This researcher is referred to as R1 in the quotes in the Results section.

- Researcher 2, an experienced educator with several years of experience in secondary STEM classrooms. She was delighted to join this project because she views criticality as an essential practice in any classroom and values the techniques this project utilized to implement it. She has expertise in qualitative research and design. This researcher also participated in the study as one of the instructors, and during data analysis, was not primarily involved in analysis of her own data until final verification of themes.

- Researcher 3, a learning and youth focused HCI researcher with experience in teaching various ages as well as leading and participating in projects foregrounding qualitative research. In the scope of the project, they joined with a deep curiosity towards the CIDER pedagogy and critical teaching more generally, and were motivated by the belief that the future of education depends on honing such critical education sooner rather than later. This researcher is referred to as R3 in the quotes in the Results section.

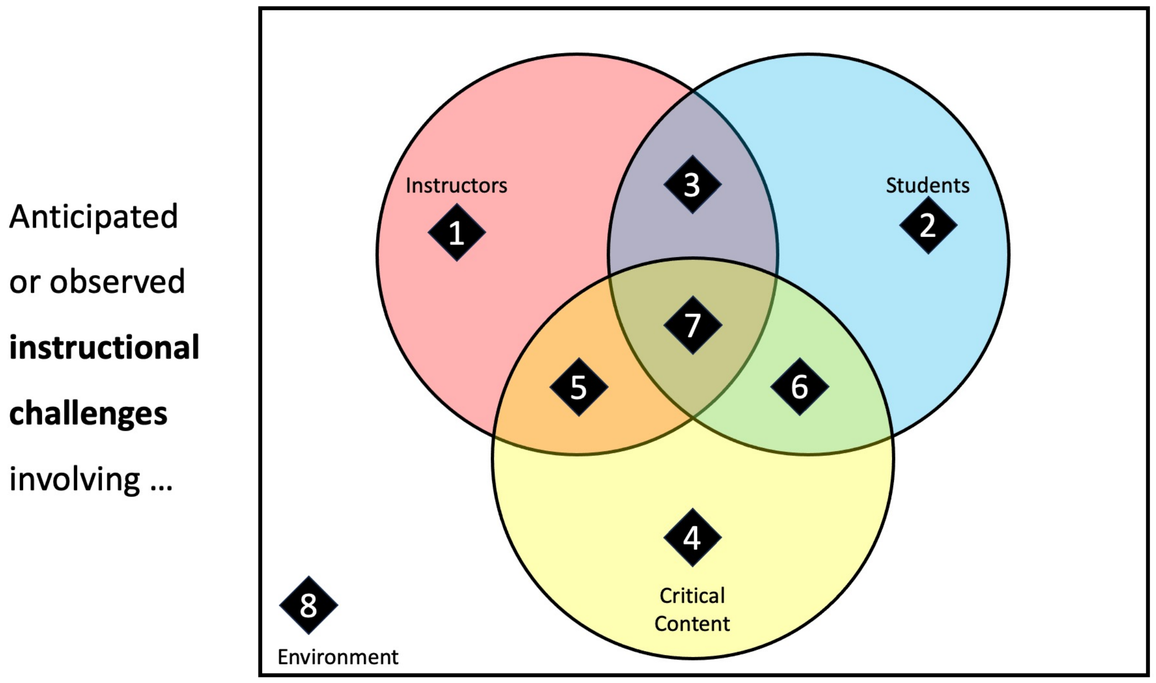


Figure 6.2: The eight overlapping categories of instructional challenge we identified through affinity diagramming, including those pertaining to instructors themselves, students, critical content, environmental variables, or some combination of the above.

To begin the inductive thematic analysis, each analyst reviewed the data that had been collected and noted down any significant insights that initially stuck out to them. We approached this initial analysis with sensitizing concepts [235] of *instructional challenges* and *mitigation strategies*, aligning with the PCK development lens described in previous sections and prior work that used these categories to understand PCK [228, 266]. Then, all three analysts met to discuss the insights they had gained and brainstorm about potential themes that described what we saw within the data.

Due to the diversity of learning contexts and the large amount of data collected, developing a categorization scheme that all analysts felt authentically represented our insights took several discussions across a series of meetings. As part of this process, one analyst (R1, the dissertation author) reviewed the textual data (interview transcripts, course observation notes, and TA support

Section	Code	Instructional challenge that involved ...
1: Instructor	Evaluation	Grading, collecting student responses to activities, assessing learning, etc.
	Flexibility	Enabling flexibility in participation styles, deadlines, etc. while encouraging engagement
	TAs	Helping TAs feel comfortable in teaching material and learning to become educators
	Time	Lack of time (to plan the course, to learn new things, to polish materials, etc).
2: Students	Motivations	Differences in motivation for taking the class, personal learning goals, etc.
	Backgrounds	Differences in prior knowledge and lived experiences
	Teamwork	Difficulties working in teams or groups
	CS training	CS students interacting with content differently than students with other backgrounds
3: Instructor + Students	Setting expectations	Clearly communicating expectations to avoid wide variations in the ways students respond to a task
	Learning moments	Helping students deconstruct problematic perspectives without shaming them for not knowing things
	Invisibility of reflection	Judging students' engagement on activities that require self-reflection, which lacks external indicators
4: Critical content	Theoretical groundings	Navigating pedagogical techniques situated in particular theoretical traditions and their limitations
	Prerequisites	Accounting for the prerequisite knowledge students need to engage with critical content effectively
	Technique mechanics	Understanding how to structure a critical design activity or adapt it to suit instructor's needs
	Reflection	Helping students reflect at the end of the activity and connect critical ideas back to their broader worlds
5: Instructor + Critical content	Confidence	Feeling comfortable with own level of content knowledge before teaching a critical topic
	Language	Using the "right" language to describe difficult critical ideas; properly defining terminology
	Positioning	Positioning and timing of critical content within the curriculum can impact how students engage with it
	Time Constraints	Navigating time constraints which necessitate decisions about what content to cover and how deeply
	Choosing examples	Choosing appropriate artifacts to critique that illustrate desired critical themes or inclusion issues
	Identity	Navigating critical content that touches upon some aspect of instructor's identities or lived experiences
6: Students + Critical content	Differing criticality	Navigating differences in students' prior experience with criticality
	Identity/agency	Students feeling that critical content is not applicable to their broader lives or future careers
	CS culture	Students more used to dominant CS cultures may conceptualize false divides between "ethics" and "CS"
	Open-ended problems	Students may struggle to engage with abstract or open-ended problems without single correct answers
	Familiarity vs fixation	Students may struggle to understand and critique completely unfamiliar artifacts, but artifacts they are too familiar with may lead to design fixation
	"Deep" critiques	Students may initially find it difficult to go beyond shallow critiques and consider design rationale
	Stereotyping	Students may voice stereotypes when thinking about how different kinds of users experience the world
7: Instructor + Students + Critical content	Resisting oppressive structures	Traditional CS educational structures are not always conducive to critical instruction/learning, so creating spaces that enable teaching of critical content can be difficult
	Navigating unpredictability	Critical instruction that allows students to bring in their funds of knowledge results in unpredictable responses, so instructors should know how to guide students in productive directions
	Inspiring hope	Navigating difficult critical ideas like oppression, bias, etc. and rebuilding senses of agency and hope
	Cultivating mindsets	Helping students gain a holistic view of critical content and develop critical habits of mind
8: External factors	Class scale	Larger classes present difficulties with student engagement, judging individual progress, etc.
	Class format	Difficulties with teaching virtual or hybrid classes; or physical lecture hall setups
	Engagement	Varied attendance and engagement with activities
	World events	Events (especially negative events) occurring in the broader world that influence instruction
	Admin constraints	Administrative constraints imposing restrictions on instruction

Table 6.5: Inductively generated codes describing types of instructional challenges anticipated or experienced by instructors and TAs, organized by the sections of Figure 6.2.

session notes) and identified examples of instructional challenges. This resulted in 285 instances of quotes exemplifying different instructional challenges.

The three analysts collaboratively affinity diagrammed these examples, arriving at an initial eight-part categorization scheme that focused on the main subject the challenge pertained to: the *instructor* themself, the *students* in their course, the *critical content* they were trying to teach, some intersection of those, or environmental variables (Figure 6.2). As part of this process, they inductively developed a set of codes for each part of the scheme describing the most common types of challenges within it (Table 6.5). The three analysts qualitatively coded each challenge according to the eight-part scheme, using the inductive codes as guidelines, allowing for multiple codes, since one quote might reference more than one type of instructional challenge. The analysts split up the work so that one analyst made the initial categorization decision for a quote and provided rationale, and the other two reviewed their decisions and noted any disagreements in their interpretation. Disagreements were discussed collaboratively by all analysts until consensus was reached.

Throughout the inductive coding effort, analysts noted some overlap in some of the challenge types across sections of the initial eight-part framework. For instance, inductive analysis of challenges in the *Section 1: Teacher* category identified some challenges having to do with a lack of time to do everything they wanted to do, but challenges involving time also manifested in the challenges coded as pertaining to *Section 5: Teachers + Critical content*. To address this, two of the analysts (R1 and R3) met to discuss these overlapping codes and develop a more refined representation of the instructional challenges, grouping the codes according to conceptual similarities.

We eventually converged on a set of eleven types of instructional challenges that instructors anticipated or observed when integrating critical computing interface design topics into their courses. Each instructional challenge in this set had two components to it: a *general instruction* component, which might manifest in any learning context, and a *critical-specific nuance* which appeared specific to computing education contexts with the teaching and/or learning of critical interface design topics. Table 6.6 shows a mapping of the initial codes to the eleven instructional challenges by the general and critical-specific components. These instructional challenges are further described in the Results section. Finally, to surface mitigation strategies for each learning challenge, one ana-

Ch-ID	Instructional Challenge	General instruction codes	Critical-specific nuance codes

1	Developing content knowledge	Confidence;
		TAs
			Theoretical groundings;
			Identity;
			Language

2A	Making curriculum decisions	Admin constraints;
		Time;
		Time constraints
			Prerequisites;
			Positioning

2B	Enabling flexible participation	Class format; Flexibility	World events
3A	Supporting engagement with open-ended problems	Open ended problems; Familiarity vs fixation	"Deep" critiques
3B	Encouraging productive peer interactions	Teamwork	Resisting oppressive structures
4A	Using a new method	CIDER mechanics	Choosing artifacts; Reflection
4B	Monitoring engagement at scale	Class scale; Engagement	Invisibility of reflection
4C	Evaluating student responses	Evaluation	Setting Expectations
4D	Addressing student misconceptions	Learning Moments	Stereotyping; Differing exposure to criticality; Navigating unpredictability

5	Understanding students' backgrounds	Backgrounds;
		Motivations
			CS culture;
			CS training;
			Resisting oppressive structures

6	Cultivating agency and hope	Identity/agency	Inspiring hope; Cultivating critical mindsets

Table 6.6: The eleven instructional challenges inductively identified during qualitative analysis, mapped onto the codes from Table 6.5 aligned with their general instruction and critical-specific components.

lyst (R1, the dissertation author) manually reviewed all of the data collected from each instructor to identify (1) whether they had experienced that instructional challenge, and (2) what strategies they had tried, if any, to address it. To further support the validity of our findings and afford instructors autonomy over how their perspectives are represented below, I also employed member checking strategies [235] once I had drafts of the insights. I describe the most salient instructional challenges mitigation strategies in the following section.

6.4 Results

Overall theme	**Challenge ID**	**Instructional challenge name**
Preparing to teach a topic	1	Developing content knowledge
Designing curricula	2A	Making curriculum decisions
	2B	Enabling flexible participation
Developing design skills	3A	Supporting engagement with open-ended problems
	3B	Encouraging productive peer interactions
Conducting interactive activities	4A	Using a new method
	4B	Monitoring engagement at scale
	4C	Evaluating student responses
	4D	Addressing student misconceptions
Accounting for student differences	5	Understanding students' backgrounds
Making content applicable	6	Cultivating agency and hope

Table 6.7: The eleven major categories of instructional challenge we identified for computing educators teaching critical interface design topics in computing courses, organized by six overarching themes of instructional activity. ID numbers correspond to the descriptions in Tables 6.8 and 6.9.

Table 6.7 describes the eleven types of instructional challenge I found during the qualitative analysis, organized by six overarching themes pertaining to kinds of instructional activity. Each instructional challenge has a general instruction component that might manifest in any kind of learning context, and a critical-specific nuance to the challenge that arose specifically when teaching and/or learning critical interface design topics within CS education contexts. Table 6.8 describes the general instructional challenges I observed organized by the challenge IDs from Table 6.7 and whether I observed that general instructional challenge code in the data from a particular instructor, indicating prevalence of the challenge. Table 6.9 indicates the same things for the critical-specific nuances around teaching and learning critical interface design topics in CS education contexts.

Consistent with the work presented in other chapters of this dissertation, I choose to adhere to the perspective on qualitative coding proposed by Hammer and Berland [132]. This means that I treat the results of the qualitative coding effort as organizational claims about themes found in the data, rather than as quantitative data in and of itself. As such, I opt not to report quantitative measures of code frequencies or calculate quantitative metrics like inter-rater reliability. Instead, I focus on providing rich descriptions of code instances that I found within the data, describing each code instance using quotes from the instructors and researchers themselves to characterize my findings.

Ch-ID	**General instructional challenge description**	**A**	**A-TA**	**B**	**B-TA**	**C**
1	Teaching a new topic without the requisite content knowledge can be difficult for instructors and TAs	✓		✓	✓	✓
2A	Pragmatic limitations necessitate that instructors must decide what ideas to cover, what to leave out, and how to structure content given time constraints	✓	✓	✓		✓
2B	Instructors who wish to afford students more flexibility can support hybrid instruction or different modes of participation, though facilitating virtual instruction presents its own challenges	✓	✓	✓		✓
3A	Students may not have been exposed to open-ended “wicked” design problems before, so instructors may need to take time to develop basic design skills	✓	✓	✓		✓
3B	Students may find it difficult to work in teams of peers	✓		✓		

Table 6.8 continued from previous page

Ch-ID	General instructional challenge description	A	A-TA	B	B-TA	C
4A	To effectively run an activity with a new method, instructors need to understand the mechanics of how it works	✓		✓	✓	✓
4B	Larger scale classes present challenges around engaging all students and effectively monitoring participation	✓	✓	✓	✓	✓
4C	Assessing student knowledge can be difficult, especially on open-ended problems or activities that draw upon students' funds of knowledge	✓	✓	✓	✓	✓
4D	Students may demonstrate misconceptions about content, which instructors can may turn into opportunities for learning	✓		✓		✓
5	Students each bring different backgrounds, motivations, and prior knowledge into the classroom	✓		✓	✓	✓
6	Students may question how course material applies to their broader lives or intended careers, and may not be motivated to engage without these connections	✓	✓	✓	✓	✓

Table 6.8: The general aspects of the eleven instructional challenges that arose when computing instructors integrated critical interface design activities into their courses, with indications of which instructor's data we observed them within. These aspects of challenges might be present in any kind of classroom. Each one is mirrored by a critical-specific nuance described in Table 6.9.

Ch-ID	Challenge nuance specific to teaching critical interface design topics in CS contexts	A	A-TA	B	B-TA	C
1	Developing content knowledge for critical interface design topics also involves understanding the limitations of theoretical traditions; reflecting upon one's own identities in relation to material; and finding the right words to describe difficult topics	✓	✓	✓	✓	✓
2A	Covering prerequisite knowledge about power dynamics and marginalization may be necessary for students to effectively engage with critical topics; How this content is positioned within the curriculum impact students' perceptions of its importance	✓	✓			✓
2B	Even with flexible participation structures, world events and external factors can still affect students' ability to participate in the kind of deep reflection needed for effective critical work			✓		✓
3A	Students who are still learning to engage effectively with open-ended problems may find it difficult to deeply interrogate assumptions and structures, falling back on shallow critiques and surface-level observations	✓		✓		✓
3B	Traditional CS education rarely incentivizes students to value peer feedback and discussion; Helping students become comfortable discussing difficult critical topics with peers takes extra time and scaffolding	✓	✓	✓	✓	✓

Table 6.9 continued from previous page

Ch-ID	Challenge nuance specific to teaching critical interface design topics in CS contexts	A	A-TA	B	B-TA	C
4A	Different example artifacts may constrain students' thinking in different ways; Without space to reflect and solidify new critical understandings, students may not make connections to other course content as easily	✓	✓	✓		✓
4B	Critical reflection and brainstorming is typically an internal process, making it difficult to monitor students' progress on critical design activities without external cues			✓		✓
4C	Critical reflection activities ask students to engage in kinds of thinking that they may not be used to and often don't have single correct answers; Particular care should be given to modeling expectations and creating learning spaces where growth is prioritized over correctness		✓		✓	✓
4D	Student misconceptions around critical topics may involve students voicing stereotypes or perspectives that they do not realize are exclusionary, inaccurate, or even offensive	✓	✓	✓	✓	✓
5	Students trained under traditional CS education structures may have particular difficulty engaging with critical topics due to the problem-solving styles and cultural values prevalent in CS spaces	✓	✓	✓	✓	✓

Table 6.9 continued from previous page

Ch-ID	Challenge nuance specific to teaching critical interface design topics in CS contexts	A	A-TA	B	B-TA	C
6	Critical activities that deal with weighty topics like oppression and bias can leave students with feelings of hopelessness; Particular care is needed to help students develop agency, hope, and habitual critical mindsets	✓	✓	✓	✓	✓

Table 6.9: The nuances of the eleven instructional challenges that arose when computing instructors integrated critical interface design activities into their courses, with indications of which instructor's data we observed them within. These aspects of the challenges are likely specific to teaching and learning critical computing. Each one is mirrored by a general aspect of the instructional challenge described in Table 6.8.

The rest of this section is structured as follows: The six themes of instructional activity listed in Table 6.7 form the basis of each subsection to contextualize when each kind of instructional challenge might arise. Each subsection then contains descriptions of one to four instructional challenges, including both the general instruction component and the critical-specific nuance that manifested within the specific contexts we studies. I characterize each component of an instructional challenge with quotes from transcripts and observation notes that describe how instructors experienced it. Finally, for each type of instructional challenge, I describe selected mitigation strategies that instructors used to address it, with indications of outcomes where possible.

6.4.1 Theme 1: Preparing to teach a topic

Challenge 1: Developing content knowledge

General instructional challenge: *Teaching a new topic without the requisite content knowledge can be difficult for instructors and TAs.*
Teaching any concept well begins with gaining a solid foundation of content knowledge for the topics involved. In larger courses with TAs, this also means ensuring that TAs have enough content knowledge to facilitate their own lessons involving this content. When brainstorming about the form of a CIDER activity, Instructor A shared that they felt this year's TAs were equipped to handle the challenge, though that may not have always been the case:

> Planning interview, Instructor A: *"I think I have more solid TAs [this year]; just a shared group of experience, a shared level of confidence and comfort with [methods like] affordance analysis*[4] *and ethical thinking has increased over the years, so I think we can be more ambitious. ... But [now] I'm just kind of reflecting on ways in which I have minimized the role of this because of structural limitations and TAs' comfort level with talking about this."*

I identified challenges around feeling confident in content knowledge or training TAs in all three main instructors' data and also in the data from Course B's TA.

Critical-specific nuance: *Developing content knowledge for critical interface design topics also involves understanding the limitations of theoretical traditions; reflecting upon one's own identities in relation to material; and finding the right words to describe difficult topics.*
When teaching critical interface design topics, developing content knowledge seemed to involve some extra steps, like reflecting on the theoretical basis of certain pedagogical techniques, interrogating one's own identities, or adapting language to avoid downplaying systemic oppression. For instance, in their debriefing interview, Instructor C and the lead researcher discussed how CIDER's roots in inclusive design traditions might have unintentionally scoped students' thinking:

[4]Affordance analysis is a pedagogical method created by Lin et al. [195] that uses the value-sensitive design framework [100] to help students critique the affordances of technologies.

> Instructor C's Debrief interview, R1: *"[CIDER] is a bit techno-solutionist in that it doesn't ask people to question explicitly whether this thing should have been designed in the first place. The idea is more 'this thing exists, let's think about how to adapt it.' Not, 'this thing exists, let's think about how to destroy it.' [...]"*
>
> Instructor C: *"Maybe spell CIDER with an S. 'SIDER.' Like 'should it exist?' Now critique it."*

To situate the usage of critical computing interface design methods within a technical course, Instructor A reflected upon their own identities and how they connected to the topics:

> Pre-teach interview, Instructor A: *"I'm also very uncertain about [affordance analysis] because I feel like I just need to have more experience with it... It's a lot of me giving my thoughts on this, connecting it to my own identity, trying to give deeper reasoning that is curriculum-related and identity-related of why I find value in this model."*

When discussing a lesson that likened web content virality and algorithmic biases to biological evolution and biological natural selection, Instructor B and the lead researcher discussed how language might impact students' conceptions of critical material:

> Pre-Teach + Planning interview, Instructor B: *"This whole video is a whole bunch of Black YouTubers talking about their experience with trying to make content be popular on YouTube ... The guy who made this video, he tried switching thumbnails to be not his Black face and he got more engagement. Whereas the general advice is 'show your face, you get more engagement.' ... So that's going back to the virality of natural selection, like societally-biased natural selection. ..."*
>
> R1: *"I wonder if you want to use the specific term 'natural selection' for that in particular. Cause *that's* racism. ... I don't think you mean to imply that's 'natural."'*

These kinds of critical-specific nuances to developing content knowledge were identified in data from all three main instructors and both sets of TAs.

Mitigation strategies: *Explicitly discussing limitations of particular design approaches; Framing oppression and bias as the result of active decisions.*

To make space for in-class reflection upon the limitations of certain design traditions, Instructor A relayed in a Slack conversation that they followed up a CIDER activity about inclusion with a movie describing the harms of dark design patterns:

> Slack, Instructor B: *"As I'm thinking about a cider activity about mental health and social media, and I'm wondering if/how dark design patterns fit in. As in, they accurately assume their audience will find loot boxes addictive, or that some subset will be. The more inclusive they are the worse it is! ..."*
>
> Instructor A: *"Yeah, I'm actually screening Coded Bias to students this week (right after CIDER yesterday) so that we get a sense of the limits of inclusive design. Making an unjust technology more inclusive is not necessarily better!"*

Following the discussion about bias in YouTube algorithms mentioned in the above subsection, Instructor B opted to change the language used in their teaching materials to frame systemic oppression as a result of active human decisions rather than a passive force of nature:

> Pre-Teach+Planning interview, Instructor B: *"I want to talk about the idea of selection. So in evolution, you have natural selection, and I'm trying to translate it into social media."*
>
> R1: *"I like the idea of [saying] 'human selection' [instead]. Because I think whenever you say something like 'natural' selection, the 'natural' part implies it's part of nature."*
>
> Instructor B: *"Normative, natural. Yeah ... I think I'm going to relabel that as as 'selection' and then have 'human selection' and 'computer selection' as subheadings. ... And when I talk about evolution initially, I'll just talk about 'natural selection'."*

While the extra effort and care to develop critical-specific content knowledge about computing interface design may have taken more time, it also enabled the instructors to have deeper, more nuanced conversations about critical design topics and share these understandings with their students.

6.4.2 Theme 2: Designing the course

Challenge 2A: Making curriculum decisions

General instructional challenge: *Pragmatic limitations necessitate that instructors must decide what ideas to cover, what to leave out, and how to structure content given logistical constraints.* In any learning context, instructors have a limited number of contact hours with students. Accordingly, they need to make decisions about what topics to cover within a curriculum and how deeply to go into each topic. Sometimes, the instructors reported that these decisions were constrained by administrative details, such as the learning technologies they had to use to report grades or communicate with students. Other times, these decisions were impacted by a lack of time for preparation. Instructor B shared that they had considered integrating CIDER into their course in a previous term, but found themself too busy with other course logistics to do so:

> Pre-Teach+Planning interview, Instructor B: *"Last time I was writing the textbook while teaching it [the course]. So I was like one chapter ahead, just trying to write the content for students to read. There were various places I wanted to put CIDER, but all I had time to do was a couple of discussion questions: 'Think about this and this and this. Next.' And I'd post it. So I didn't get a chance to really sit down and think about how CIDER works."*

Instructional challenges involving time and administrative constraints arose in all three main instructors' data, as well as the data from Course A's TAs.

Critical-specific nuance: *Covering prerequisite knowledge about power dynamics and marginalization may be necessary for students to effectively engage with critical topics; How this content is positioned within the curriculum impact students' perceptions of its importance.*

Because post-secondary computing programs tend to focus on developing technical skills and many general education courses do not include critical instruction, many computing students may never have been exposed to concepts like systemic oppression, hierarchies of power, and disproportionate impacts of technological biases in formalized education (though they might very well have lived experience with them). As a result, when teaching critical interface design topics in computing courses, instructors should be aware of the need to develop that foundational knowledge themselves. In their debriefing interview, Instructor C reflected upon the need to help students reflect upon the implicit power structures and stereotypes in CS culture and education before they could productively engage with critical interface design ideas:

> Debriefing interview, Instructor C: *"More than half the class identifies as men, which is the number one population that teaches CS. Even though some ridiculous percent of [general] teaching is women, there are a lot more men than women in this cohort. [...] Some of them hadn't done some of the interrogation around gender and CS yet, and who 'belongs' and who 'doesn't'. Like, the cultural layers that we have [to deal with] before they even get to us. We're not fighting this in the classroom, we're fighting this in the cultural [sense], if that makes sense?"*
>
> R1: *"So without that kind of interrogation done, how do you think that impacted things?"*
>
> Instructor C: *"I think it was like—We did that interrogating the first 8 weeks, and then, on week 9, we had done the interrogating [needed] to do CIDER."*

Instructors also noted that decisions about when and how to introduce critical topics within the curricula could implicitly signal notions of value to students, especially if it is positioned differently from technical content. Instructor A's critical interface design lessons fell on particular days of the week due to their course schedule, and they reflected upon what that meant in the context of the course:

> Planning interview, Instructor A: *"There's actually a way in which I am subconsciously minimizing the place in the curriculum of these 2 [critical content] ideas, because I put them on Friday and Monday. And specifically on Friday two weeks from now - because we have [lab] Section on Thursday. You spend the first Monday and Wednesday on like 'actual' materials [makes air quotes]. Binary heaps, hash tables. And then you have Section that practices that. [...] We have always had 'gentle' content on Friday."*

Critical-specific nuances around prerequisite knowledge and content positioning were observed in data from Course A (mentioned by both Instructor A and the Course A TAs) and Course C (mentioned by Instructor C), but not in data from Course B.

Mitigation strategies: *Scaffolding critical content knowledge before asking students to apply it; Affording enough time for students to process initial emotions.*

To ensure that their students had the prerequisite knowledge needed to effectively engage with CIDER activities, Instructor C positioned their activity toward the end of the course, and spent preceding weeks developing students' understandings of different "lenses" of accessibility and inclusion through which they could critically reflect on interface inclusion:

> Pre-Teaching interview, Instructor C: *"The first week we talk about the AP Exam, the second week we talk about standards, and then, after that, we just look at lenses. [...] Week 4 is creativity, Week 5 is types of grading, Week 6 is self efficacy. Then we've got UDL[5]. So those kinds of lenses, and then week 9 [with the CIDER activity] it's a good wrap up, like, 'Are you applying all the lenses?"'*

Some instructors also underscored the importance of allotting course time for students to process emotions when dealing with weighty topics. When asked in their Post-Teaching interview what advice they would give a peer or colleague who wanted integrate critical interface design topics into their course, one of Course A's TAs responded:

[5]Universal Design for Learning, a framework of how to make learning materials accessible to learners with varied capabilities. More on UDL can be found at `https://udlguidelines.cast.org/`

> Debriefing+Post-Teaching interview, A-TA1: *"When we did the activity with our staff team [as a practice run], they spent a good chunk of the time just talking about misinformation, and their thoughts on that and their experiences with it [...] So, [I'd suggest] having space in the lecture for students to talk about that before we went into the time for the activity. I think that's really important, especially with anything that's remotely polarizing, or makes you feel things, just so people can confront their feelings and then be ready to think critically."*

Though instructional time is nearly always constrained in post-secondary courses, spending the extra effort to create instructional spaces conducive to critical reflection seemed to help these instructors more effectively teach critical and inclusive interface design topics.

Challenge 2B: Enabling flexible participation

General instructional challenge: *Instructors who wish to afford students more flexibility can support hybrid instruction or different modes of participation, though facilitating virtual instruction presents its own challenges.*
Especially after the increase in remote and hybrid learning necessitated by the COVID-19 pandemic, offering students flexibility in how they attend and engage with course sessions has become more and more prevalent in post-secondary computing education. Flexible participation can support students in learning in ways that best fit into their broader lives, though it also presents its own instructional challenges for instructors to navigate. Instructor B mentioned the tradeoffs of providing flexible participation options while trying to run interactive activities in their hybrid course:

> Post-Teaching interview, Instructor B: *"I want flexibility AND people to show up [in person]. [...] I erred on the side of flexibility and people didn't show up. [...] I also had trouble [with], like, if there's a 'discuss something with your neighbor' thing, that worked well in-person. It did not work well on Zoom. I think there might be ways of getting TAs to manage that on Zoom, or try to do that, but that takes quite a bit of extra effort to try to make discussions happen."*

Additionally, when running in-person courses, an inflexible physical setup of the room might constrain instruction. Instructor A shared how the layout of their assigned classroom partially dictated decisions about group size:

> Planning interview, R1: *"How big are the teams?"*
>
> Instructor A: *"Two to three people [...] And that's partly constrained on the way the lecture halls laid out, where it's like you're in rows, so you can only turn to partners. And that way you can still be seated together and actually talking."*

Instructional challenges around class format and flexibility were identified in all three courses' data, with all three main instructors mentioning or experiencing them, as well as Course A's TAs.

Critical-specific nuance: *Even with flexible participation structures, world events and external factors can still affect students' ability to participate in the kind of deep reflection needed for effective critical work.*

As mentioned throughout this section, critical interface design ideas can be difficult for students to grapple with. Students might struggle to learn *any* new ideas when they are under stress from other academic, family, work, or life situations [311]. These difficulties are even more pronounced when reflecting upon critical ideas that deal with weighty topics like bias, oppression, and technological harm. Instructor C, who had integrated a CIDER activity into one previous offering of the course prior to the study, relayed how they adapted their instruction on that day to account for traumatic events occurring in the news:

> Pre-Teaching interview, Instructor C: *"[On the day we did CIDER] That was also the day we had a shooting. When you're working with teachers when there's a mass school shooting, you're just like: 'What do you need today? Cool, cool, cool. You need an hour early release? All right. We're gonna do that. You go spend time with your kids', cause a lot of them had kids and family."*

These kinds of critical-specific instructional challenges around world events were explicitly identified in two of the three courses, in data from Instructor B and Instructor C.

Mitigation strategies: *Creating learning environments conducive to critical learning.* To ensure create learning environments where students felt supported and safe engaging in critical work, Instructor C explicitly designed their class environment to help students feel like they could bring their full, authentic selves into the space:

> Debriefing interview, R1: *"You've really created a space where you're normalizing things that are typically considered abnormal in a classroom. I noticed you called out executive function and anxiety and deadlines and things like that. [...]"*
>
> Instructor C: *"That is my normal classroom. Welcome."*
>
> R1: *"Do you feel like you'd be able to get students talking and critiquing and thinking in these critical ways if you hadn't created that sort of culture in the classroom? Or do you feel like that's an integral part of tackling these tough, critical computing concepts?"*
>
> Instructor C: *"I think it's important. You've got a lot of stuff to disassemble in order to have a classroom where students feel safe not meeting your expectations, but actually saying what they think and feel. [...] You have to create a classroom where students can engage critically and say what they really think versus what they think you want to hear. So their grades need to not be on the line, participation needs to not be on the line."*

Prior work on teaching critical computing topics suggests co-constructing these kinds of classroom spaces is key students feel comfortable tackling tough topics [92].

6.4.3 Theme 3: Developing design skills

Challenge 3A: Supporting engagement with open-ended problems

General instructional challenge: *Students may not have been exposed to open-ended "wicked" design problems before, so instructors may need to take time to develop basic design skills.*

Within prior work in design and HCI education, students are known to find it initially difficult to engage with open-ended design problems, especially those that do not have single, simple correct answers [39, 268]. Some prior work speculated that within computing education contexts, this difficulty may be due to the ways that technical courses train students to problem-solve, which is at odds with the messiness of open-ended, complex design problems with many moving parts [229]. One of course A's TA's identified this type of instructional challenge:

> Debrief+Post-Teaching interview, A-TA2: *"When people have a really action-oriented mindset, which is common for technical people, [...] It's hard to engage with speculative activities. Cause they're like, 'Oh, this makes me uncomfortable. There's no right answer.' I'm interpreting a lot of what I think students are thinking. But it's really outside of people's comfort zones to not have a defined answer, or also just getting comfortable with the fact that there's big wicked problems, and we can't probably solve them. But we can still talk about them. But people sometimes don't want to talk about problems that they feel like they can't solve, and that's uncomfortable."*

Design fixation can also be an issue with students who are not used to designerly brainstorming, especially when students are primed with artifacts that they are overly familiar with [219, 229]. Many of Instructor B's CIDER activities focused on social media that students were already familiar with, which may have caused students to overlook assumptions that Instructor B thought were more evident:

> Post-Teaching interview, Instructor B: *"One of the things that stood out to me is the assumption that the naming policy for Facebook assumes that the account is owned by an individual. [...] But students really didn't didn't jump on that. [...]"*
>
> R1: *"You're right, that does feel like a pretty poignant assumption that they could identify."*
>
> Instructor B: *"Yeah, but it's also not—it's an assumption that's everywhere. So it's really hard."*

> R1: *"Yeah, absolutely, [when] you're in the water of social media, it's hard to recognize it."*

General instructional challenges involving open-ended problems and design fixation were observed in all three courses, within data from all three main instructors and Course A's TAs.

Critical-specific nuance: *Students who are still learning to engage effectively with open-ended problems may find it difficult to deeply interrogate assumptions and structures, falling back on shallow critiques and surface-level observations.*
Students who struggle with the above basic design skills may have an even harder time engaging deeply enough to critique rationale and surface assumptions. In the courses I observed, this led to some students shallowly critiquing the interfaces during CIDER activities, not quite getting to the level of interrogation of power structures that instructors desired. For instance, while going over student responses to their first critical interface design activity, Instructor B and R1 noted that fewer students responded to the CIDER questions that asked them to consider design rationale:

> Debriefing interview, Instructor B: *"I got 35 responses to that [generic question about online privacy]: 'How do you feel about this tracking?' And then the CIDER responses were a lot less. [pulls up responses on computer] 'What critiques and assumptions did you come up with?' [...] Only 5 people that came back with something. [...]"*

> R1: *"And only one of those is actually an 'assumption'. [...] I wonder if it's just easier to critique something rather than to frame it as an assumption."*

I identified instructional challenges around depth of critique in all three courses, from data from all three main instructors.

Mitigation strategies: *Framing collaborative reflection as an actionable method for open-ended problem-solving; Helping students identify artifact purpose before diving into critique.*
To help students gain confidence working with open-ended problems, Course A's TAs positioned the peer feedback aspects of CIDER as a way to collaboratively address open-ended critical problems that resist simple solutions:

> Debrief+Post-Teaching interview, A-TA2: *"[Misinformation] is a really complicated problem that needs a lot of brains looking at it and iteratively discussing solutions [...] Students are saying, 'I can't do this. This is the first thing I ever learned about misinformation.' Maybe you alone in the silo can't solve this, but if we have this nice [CIDER] framework for collaborating, maybe we can be more productive."*

Instructor C also noted the importance of ensuring students understood the purpose of an artifact before they began attempting to make it more inclusive. During their course observation on the day that Instructor C ran a CIDER activity, R1 observed that the instructor had students reflect upon the intent of a test question before they began critiquing it. When asked about their approach, Instructor C responded:

> Debriefing interview, Instructor C: *"One thing I surfaced in this particular case that might be interesting in CIDER broadly is the need for [students to understand] purpose. Like, 'what is the purpose of the keyboard?' [...] It's the 'what are we trying to do' part of the brainstorm. We could redesign a test question to ask another question. But are we still getting that initial purpose of that question? So when we're redesigning, how do we make sure that [we preserve] the initial intent of that question?"*

Each of these strategies seemed to help instructors get students feeling more comfortable working with the nuances of open-ended problems that often arise in critical design work.

Challenge 3B: Encouraging productive peer interactions

General instructional challenge: *Students may find it difficult to work in teams of peers.*
Working in teams or groups is also known to be an initial learning barrier for students, especially when the need to socially navigate team dynamics overlaps with the learning of new difficult content knowledge [229]. Instructor B ran a mid-quarter anonymous feedback survey on the course and asked students their thoughts on the different kinds of peer discussion activities they had integrated into the course. One student responded that they preferred individual work:

> Debriefing interview, student in Instructor B's course: *"'I enjoy them, but I also have no friends in the class, so I don't know anyone; would rather just participate on my own than be forced to discuss with all these students."'*

Instructional challenges around facilitating team and group work arose in two of the three courses, within data from Instructors A and B.

Critical-specific nuance: *Traditional CS education rarely incentivizes students to value peer feedback and discussion; Helping students become comfortable discussing difficult critical topics with peers takes extra time and scaffolding.*
When it comes to critical instruction within computing education, instructors encountered some unique challenges that accompanied getting computing students comfortable with working in groups. For instance, instructor A observed that traditional CS education typically incentivizes individual work over collaborative efforts, whereas CIDER does the opposite:

> Planning interview, Instructor A: *"[We're] looking at: How do we engage and work with the community? How do we actually design for community rather than assuming that we know best? [...] A lot of classes in computing education are about empowering students to individually do something, without talking to other people right? So there's a whole kind of paradigm shift there [in the CIDER activities] that I like. I think we're not really having this conversation [yet] in CS."*

All three instructors and TAs mentioned instructional difficulties around resisting the normative structures of CS education that may have led students to devalue the importance of teamwork and discussion around critical computing interface design problems.

Mitigation strategies: *Allotting time to develop students' teamwork skills before introducing critical group discussions.*
To help students get to the point where they could discuss have deep, nuanced critical group conversations, Instructor A explicitly allotted time for students to get used to working with teams before asking them to engage in shared critical reflection:

> Planning interview, Instructor A: *"Rather than throwing all the stuff at them in the beginning [and] expecting them to pick it up all at once, I'm saying this first two weeks is just onboarding. Get used to working with a team in class, that's already hard enough. And then we'll add on roles for making your team work better, and then we'll add on [critical interface design activities] afterwards. So you can actually work on more social challenges as well, in addition to the technical challenges which we are used to doing in CS. [...] In the past they had students at the beginning, from day two, learn how to be a team AND start to do roles AND still learn new content. I think that's a lot to do all that effectively at once."*

Instructor A's layered approach seemed to be effective, as they described in later interviews having relatively high student engagement with their CIDER activities.

6.4.4 Theme 4: Conducting interactive activities

Challenge 4A: Using a new method

General instructional challenge: *To effectively run an activity with a new method, instructors need to understand the mechanics of how it works.*
As with any new teaching technique, instructors need to feel confident in the details of how it works before they use it in their course. Accordingly, the instructors involved in the study asked several clarifying questions around the mechanics of the CIDER technique as the study began, including what the different stages involved, how to effectively facilitate activities, and how to collect students' responses. Some noted how CIDER's focus on interactivity and peer discussion differed from more "passive" lecturing styles. When asked what advice they would give to a peer or colleague interested in integrating CIDER into their course Course B's TA noted that the activity's success would depend partially on the instructor's comfort with running these kinds of activities:

> Debrief+Post-Teaching interview, B-TA: *"You have to ask yourself, how much do you want students to engage in this? Because the traditional college [lecturing] model is*

> *very passive learning, and I think CIDER is a very active one. I think the question I would ask that person is, how much can you facilitate active discussion? Why do you want that? And how can it help your students? And then being like, okay, now we can talk about CIDER."*

Instructional challenges around method mechanics were observed in data from in all three courses, by all three main instructors as well as Course B's TA.

Critical-specific nuance: *Different example artifacts may constrain students' thinking in different ways; Without space to reflect and solidify new critical understandings, students may not make connections to other course content as easily.*
Prior evaluations of the CIDER technique [226] suggested that the type of artifact used as the basis for the activity may have guided students' reflection. I shared this insight during instructors' planning interviews, which led to challenges around how to choose artifacts that would most effectively illustrate the kinds of critical interface design issues instructors wished to target. For instance, Instructor B discussed difficulties selecting an appropriate artifact for a lesson on data privacy.

> Pre-Teach+Planning interview, Instructor B: *"[I want students] to think about who this [privacy framing] makes sense and for who it doesn't make sense for. [...] I'm not sure what the artifact is for this yet, but I'm wondering if there's a privacy settings screen, or privacy rules or something. [...] I want to give students a chance to try to think about what privacy means and settings that aren't necessarily the default. So I don't know an artifact for that yet."*

Further, the CIDER framework does not contain an explicit reflection component to conclude the activity, leaving the choice of how best to connect new understandings to course content up to educators for maximum flexibility. Instructor A, whose curriculum contained low-level technical content, opened a discussion about overloaded terminology in the community Slack channel with a series of messages about the word "assumption":

> Slack message, Instructor A: *"We talked about ensuring that we defined 'assumptions' (in inclusive design) as different from an assumption in mathematics (as in, 'assume X'). I'd love to hear some possible definitions for 'assumption' in the context of inclusive design."*
>
> Instructor A: *"Maybe 'statements about users that designers take as true without proof'. I think the focus on users is interesting because it suggests that designers take assumptions about users and then encode them into the design of products."*
>
> Instructor A: *"This would suggest that addressing assumptions about users can lead to differently-designed products."*

The conversation continued with other community members contributing their own definitions and noting that though the words were the same, there was certainly a distinction to be made. I observed instructional challenges related to choosing between different artifacts and framings to guide critical reflection processes in data from all three courses, including the three main instructors and from Course A's TAs.

Mitigation strategies: *Redefining overloaded terms to highlight connections to critical content; Choosing 'right-sized' artifacts for students to critique.*
While the overloading of certain design terms with mathematical or technical terms did cause Instructor A some initial difficulty, they later described how they leveraged the overloaded terminology as a way to help students connect critical content back to the technical ideas:

> Planning interview, Instructor A: *"I think that [overloading] actually works well with the way I teach this class because I am providing more expansive definitions for things. Like, how you analyze a program can be more expansive than just runtime. [...] we're talking about trade-offs, not only trade-offs in terms of design from a runtime perspective, but also in terms of what you choose to implement, the features you choose, which then have a downstream impact on the ease of implementing, ease of maintenance, [and] broader impacts as well as efficiency."*

To address the challenge of choosing appropriate artifacts, Instructor B reasoned through the nuances of choosing an artifact with enough depth for students to identify several different embedded assumptions about data privacy, but that was still short enough for students to understand it within limited course time:

> Pre-Teach+Planning interview, Instructor B: *"If there is a good GDPR page or summary, that might be a really good artifact to look at. It's hard to know without actually looking at it if it's going to work [... If it's] 50 pages, there's no way we're going to do that as an in-class activity. Unless the critique is 'well, no one's going to read that."'*
>
> R1: *"[Spends several minutes looking at different GDPR summaries with Instructor B] This one is 'What is the GDPR'. Here's a few bullet points, and then it's like 'use of personal data', or like 'must be respectful to the individuals.' ... It is so individual focused."*
>
> Instructor B: *"Yes, and that's one example [of an embedded assumption]. Maybe there are others! I don't know. I'm hoping there's more than one answer. If there's just one answer, I don't think it makes a good CIDER activity."*

Instructor B ended up using the individual-focused summary of the GDPR to good effect in their course, noting in later interviews that short summary did seem to enable understanding without overwhelming students.

Challenge 4B: Monitoring engagement at scale

General instructional challenge: *Larger scale classes present challenges around engaging all students and effectively monitoring participation.*

Encouraging engagement with course material and monitoring student engagement can be difficult at the best of times. These challenges are exacerbated as course enrollment scales up, since educators can no longer engage with each student to understand and adapt to their individual learning progression. Even within their relatively small course, Instructor C encountered issues with scale and engaging each person in the course:

> Planning interview, Instructor C: *"The class is big enough that I don't know what everyone's thinking this year."*

Instructional challenges around class scale were noted in all three courses, by all instructors and both sets of TAs.

Critical-specific nuance: *Critical reflection and brainstorming is typically an internal process, making it difficult to monitor students' progress on critical design activities without external cues.* When it comes to learning activities that involve critical reflection in particular, engagement is particularly difficult to monitor because individual reflection rarely has externally evident cues. The CIDER technique's different stages ask students to brainstorm critiques of artifacts on their own before transitioning into peer feedback discussions. This kind of individual reflection work can present challenges to knowing whether students are engaging with content or just "waiting out the clock", as mentioned by Instructor B:

> Post-Teaching interview, Instructor B: *"The amount that people are willing to just sit and wait out the clock is a problem. In any activity where they have to think on their own, it's hard for me to know if they're thinking. I can't see inside their head. I don't know if they they need more time or if they're bored. I don't know I what's going on. I get self-conscious and worried that they're bored because I'm just sitting there waiting out the clock."*

Instructor B and Instructor C both noted instructional challenges around the invisibility of critical reflection.

Mitigation strategies: *Using external progress indicators to gauge critical engagement.* To address their concerns about the invisibility of critical reflection and engagement, Instructor B speculated about the utility of having students physically write out their responses to CIDER activities rather than use their devices to submit responses. They also noted that externalizing their ideas might help students sharpen critical insights:

> Debriefing interview, Instructor B: *"I should really bring paper and pens and pass those out [...] Give them something besides just their phone to engage with if they're trying to brainstorm."*
>
> R1: *"How do you think it would help with engagement to have that physical medium?"*
>
> Instructor B: *"So, I'm projecting here. But for me, if I'm just thinking in my head, I don't have to be as concrete. [...] I can think hazily. If I'm trying to write something down, that forces me to be more concrete. [...] But the other thing is for me as a teacher, if I see students writing, that also tells me what they're doing."*

Course B's TA confirmed that Instructor B had implemented an optional pen-and-paper response method in later offerings of the course where they used the CIDER activity, and that it did seem to help with student engagement and classroom monitoring. Instructor C employed a similar strategy to monitor students' critical reflection, using a shared Google Slides deck and encouraging students to open it in a private browsing tab so they could still share anonymous reactions:

> Planning interview, Instructor C: *"I do everything in shared Google Slides, so they're collaborative. [...] Last week we were talking about identity and belonging in CS. And so I gave each [Zoom breakout] room an academic paper to read, and then they had to synthesize it and turn it into a slide, and then come back. And we did a 'gallery walk-through', which just means you have 5 min to read everybody's slides. [...] Sometimes I'll be like: 'All right. Everybody put this in an incognito window because I don't want to know who you are, and I want you to feel free to respond."'*

Instructor C's usage of shared slide decks also enabled students to see their peers' responses to critical reflection activities in the gallery walkthrough, providing different perspectives for students to engage with.

Challenge 4C: Evaluating responses

General instructional challenge: *Assessing student knowledge can be difficult, especially on open-ended problems or activities that draw upon students' funds of knowledge.*
Under traditional academic structures, educators typically have to assign grades to students, which means they must come up with some way to evaluate students' learning. As a result, several instructors asked for support in interpreting and evaluating students' responses to the CIDER activities they ran, especially around what kinds of things they should look for when grading. In particular, Instructor A noted the difficulty inherent in judging the quality of a response that drew on a student's lived experience rather than facts about the world:

> Planning interview, Instructor A: *"That's [Grading is] an interesting one, because we're talking about students, and especially if we're talking about agency too—What is the need to be able to bring in your funds of knowledge? And we actually judge that [with a grade], is that actually a just thing to do to judge someone's contribution?"*

All three main instructors and both sets of TAs mentioned or experienced instructional challenges around evaluation.

Critical-specific nuance: *Critical reflection activities ask students to engage in kinds of thinking that they may not be used to and often don't have single correct answers; Particular care should be given to creating learning spaces where growth is prioritized over correctness.*
When students are assessed through closed-ended test questions that have single correct answers, evaluation is often easier. With more open-ended activities that do not have single correct solutions, assessing students' learning can be difficult, since they might not yet understand how to engage with these kinds of problems. When asked what advice they would give to a peer or colleague interested in teaching with critical interface design topics, Course A's TAs also noted the need to set clear expectations around how students should engage with critical content:

> Debrief+Post-Teaching interview, A-TA2: *"Have a clear expectation [set for students]. Otherwise, I feel like people just sort of spiral because they're like, 'I don't*

> *know what I'm supposed to do."'*

> A-TA1: *"You get a much wider range of responses, I think, where some people will take it and run with it, and then others kind of do the bare minimum."*

Further, as noted previously, students trained under educational structures that value correctness over growth might also be hesitant to engage in fear of giving "wrong" answers. Instructors trying out new kinds of critical pedagogical techniques face the potential risk of their first few activities not going as smoothly as they had hoped. In these cases, Instructor C noted the challenge in ensuring that pedagogical experimentation does not make students feel that *they* were in the wrong:

> Planning interview, Instructor C: *"[My instructional approach] is going to be specific to each group of people I have. Every year's a little different [...] I'm designing this year's a little different than how I'm designing last year. How I'm talking about assignments this year is a little different than how I talked about assignments last year. [...] Sometimes it's an objective failure. And I'm like 'oh, that's on me, pass, y'all get an A, we'll do it again next week.' Or like, 'this doesn't count for credit. Thank you for trying,' you know, owning it. And making sure that the only person that suffers consequences from that is me, as much as possible."*

I identified instructional challenges related to setting expectations and grading critical learning activity in all three courses, in the data from Course A's TAs, Instructor B, and Instructor C.

Mitigation strategies: *Providing formative, reflective feedback on critical tradeoffs; Modeling how to engage with critical activities.*

To navigate the nuances around grading open-ended responses to critical design activities, Instructor A opted to give credit for participating, then had TAs follow up and comment on submissions with prompts for students to think deeper about content:

> Debriefing interview, Instructor A: *"For the assessment part, it's a little bit easier for me because I just have to say: 'Did you do it or not?' Where the final mark that we*

> *would have to give them is a checkmark or an X/resubmit kind of thing. So [other aspects] could just then be more focused on TAs giving just helpful feedback on ways students might be able to expand their thinking [...] Like, 'There's lots of [design] options here. How do you weigh the different choices?' and just to get the TAs to prod that a little bit [...] Even just kind of raising it as this kind of decision point you're going to be making. And you're really looking at the values that you have, the values of your company, your team, on what we're doing, what we're prioritizing here."*

Modeling activities can also help reduce variability in responses, making it easier to understand students' learning progressions. During their course observation, R1 noted that Instructor C had students practice the CIDER framework once together as a class using a desktop keyboard before releasing them to in groups to critique AP CS assessment questions. When asked about this approach, Instructor C noted how modeling behavior could help students feel more confident engaging in critical design activities:

> Debriefing interview, Instructor C: *"We're all good at concrete things, and an [assessment] item is a little bit more abstract. I learned last year that you have to model and like, when you have a concrete thing [like a keyboard] that everybody had to use to get here [to virtual class] today, it was easier to have that [modeling] conversation quickly [...] They were participating. But I was also guiding."*

The mitigation strategies employed by these instructors helped them navigate the instructional challenges involved with evaluation of critical interface design learning.

Challenge 4D: Addressing misconceptions

General instructional challenge: *Students may demonstrate misconceptions about content, which instructors can turn into opportunities for learning.*

In any learning context, students might develop or bring in misconceptions about course material, which may become evident during instruction. When these misconceptions become evident, it presents a “learning moment,” to use the words of one of the instructors: an opportunity for instructors to help students correct their understandings of material. To best support learning, care should be given to helping students address misunderstandings without making them feel ashamed of mistakes or for not knowing content. Instructor A spoke to the challenges of even surfacing learning moments in larger classes, where students might not speak up for fear of being wrong in front of peers:

> Planning interview, Instructor A: *“When you’re contributing ideas [to] the broader public, whether it’s the class, or anyone else on your team... It’s cool to have that kind of [anonymity]. Yeah, it’s not like, ‘I’m the one who came up with this idea, and therefore I can be shamed for it.”’*

Other instructors also noted particular difficulties around encouraging students to engage in speculation around topics they didn’t yet feel confident in. I observed instructional challenges related to supporting effective learning moments in all three courses, in data from all three main instructors.

Critical-specific nuance: *Student misconceptions around critical topics may involve students voicing stereotypes or perspectives that they do not realize are exclusionary, inaccurate, or even offensive.*

As mentioned before, the topics involved in critical computing interface design instruction inherently connect to the broader world. CIDER activities, in particular, ask students to both leverage the funds of knowledge [278] they have gained through their lived experiences, and to speculate about how people with different capabilities, contexts, and access to resources might be impacted by particular design assumptions. As such, students engaging in activities that ask them to consider the experiences of different kinds of people may hold or voice harmful, inaccurate stereotypes during the activity. Instructor B relayed challenges getting students to consider different kinds of users beyond “tech dudes”:

> Pre-Teaching interview, Instructor B: *“I really want them to be thinking about a lot of different users of these social media systems. I want them to get outside of the box of just themselves as users. I feel like Tech has a really big problem with all the dudes*

> *in San Francisco making programs for dudes in San Francisco, and they aren't really thinking outside of that."*

Additionally, students engaging in activities that ask them to consider the experiences of different kinds of people may hold or voice harmful, inaccurate stereotypes during the activity. Instructor C noted the inherent difficulty in navigating problematic misconceptions when they did arise in the course of instruction:

> Pre-Teaching interview, Instructor C: *"Sometimes I had to run a little interference [during class]. ... it's an interesting spot to be in, the position of the instructor, because you don't want to shame students, but you don't want other students to think you agree with that student?"*

Critical-specific nuances to the challenge of addressing misconceptions were identified by in all three courses, mentioned by all three main instructors and both sets of TAs.

Mitigation strategies: *Addressing potentially problematic perspectives directly; Helping students reflect upon and interrogate their own assumptions.*
Instructor B used a learning technology for some of their critical design activities that allowed students to respond anonymously and have those responses projected to the class in real time. When an anonymous student submitted a response containing a reference to online content known to be associated with alt-right white supremacist political activity, Instructor B took it as an opportunity to discuss dog whistles[6] on social media:

> Debriefing interview, Instructor B: *"I watch [responses] come up live and there's been a couple of times where I've commented, added some context to something. Like, someone said that they enjoyed Pepe the frog memes—Pepe the frog is a comic, but it's been largely overtaken by alt-right neo-nazis. It's not how it's always used. But there's this thing where neo-nazis are sort of trying to overtake this."*

[6]From Wikipedia: "In politics, a dog whistle is the use of coded or suggestive language in political messaging to garner support from a particular group without provoking opposition. The concept is named after ultrasonic dog whistles, which are audible to dogs but not humans. Dog whistles use language that appears normal to the majority but communicates specific things to intended audiences." [301]

> R1: *"So it's not always, but it can be a dog whistle."*
>
> Instructor B: *"Yeah. So I tried to not directly attack the student or say what the student's motivation was. It was anonymous. But also not just let that [response] slide as normal."*

On the other hand, Instructor C noted that students sometimes have good intentions, but just haven't had the space to critically reflect upon their own assumptions and misconceptions about how to solve problems. In these cases, Instructor C used a series of questions to help students build new critical understandings:

> Post-Teaching interview, Instructor C: *"I think sometimes [students have] the best of intentions, like, 'Oh, let me solve this problem that I imagine these people have.' [...] One of the students was like, 'I'm going to use AI to help all students with disabilities.' So I was like, 'Oh, do you have any disabilities? Which disabilities [do you want to address]? Do you know anybody with that?' And so, you can kind of start to ask, 'Do you think you can design for them, or do you think they would be interested in designing for themselves?' You can ask questions that get you there without shaming students."*

While misconceptions around critical interface design topics might be socially and politically fraught, addressing stereotypes and problematic perspectives directly can help students build new critical understandings without feeling shamed for not knowing material.

6.4.5 Theme 5: Accounting for student differences

Challenge 5: Understanding students' backgrounds

General instructional challenge: *Students each bring different backgrounds, motivations, and prior knowledge into the classroom.*

In any course, students come to the classroom with differing personal and educational backgrounds.

Students' motivations for taking a course also differ, ranging from genuine interest in the content to simply fulfilling a program requirement, as Instructor A noted when reflecting on their course design:

> Post-Teaching interview, Instructor A: *"In all of this is this question of: To what end are we learning all this material? And in what ways can I address that? And I think students are already here [gesturing], I think students are already taking my classes for a variety of reasons, for diverse reasons. And the question is, how can I better serve those reasons?"*

All three main instructors identified instructional challenges around navigating individual student differences in motivation and prior knowledge, as well as Course B's TA.

Critical-specific nuance: *Students trained under traditional CS education structures may have particular difficulty engaging with critical topics due to the problem-solving styles and cultural values prevalent in CS spaces.*

Traditional computing education courses rarely teach critical content, so students may enter a course with differing levels of exposure to criticality and inclusive design experience. For instance, Instructor C noted differences in the level of readiness students showed in being motivated to discuss critical topics from a previous year's course:

> Planning interview, Instructor C: *"It has been more work this quarter than it was last year. Last year it was like, 'oh, yeah, [we're] immediately on the critical bandwagon' and this quarter they are not immediately on the critical bandwagon."*

Some instructors noted that specific aspects of computer science culture could make it more difficult to teach critical topics. One of Course A's TAs described difficulties in resisting dominant CS cultures that might lead students to believe that there is no place for critical content in computing courses:

> Debrief+Post-Teaching interview, A-TA2: *"It's sort of a problem that I've observed in technical classes, when you start asking people to engage with justice-oriented con-*

> *tent. Sometimes there's a little bit of friction, like 'why would we do that, this is a coding class.' [...] Or they're like: 'this is pointless,' because maybe we can talk about solving problems all day here in our ivory academia tower. But if I go into my industry internship and they're just like: 'Code some stuff for the Defense Department' or whatever, I don't feel like I can say anything about it."*

I observed these kinds of instructional challenges in data from all three courses, with all three main instructors and both sets of TAs noting these kinds of critical-specific nuances around the difficulty of teaching critical topics in CS classrooms.

Mitigation strategies: *Lowering gatekeeping barriers by making content relatable; Leveraging technical identities to motivate critical content.*
To make critical computing concepts topics more approachable for students who were less confident in their technical identities, Instructor C explicitly broke down terminology around computing concepts (in this case, HCI) by connecting it to students' everyday lives and experiences:

> Course C Observation notes, R1: *"[Instructor] mentioned that CIDER is designed to be used w/ HCI students – then defined HCI as any time humans and computers interact"*

> Debriefing interview, R1: *"I also noticed that you did call out explicitly that CIDER is [originally created] for HCI students. And then you made HCI a thing that people do every day. Can you speak to some of your thoughts there?"*

> Instructor C: *"One of our goals in this broader program is to break down the gatekeeping of CS. [...] Computer science has an insecurity problem, and so there's a lot of gatekeeping about everything. So I'm like, 'anytime you interact with the computer— You're human interacting with a computer. That's what HCI is. And this is how we study it.' So I think it was dismantling gates."*

This kind of anti-gatekeeping rhetoric might help students feel more equipped to tackle the technical aspects of critical computing interface design work. On the other hand, for students who were more confident in their technical identities, Instructor A leveraged students' CS backgrounds as a means of increasing students' confidence for solving difficult critical problems. They shared how they framed their critical interface design activities as opportunities for students to showcase the depth of their technical knowledge and connect it to the broader world:

> Planning interview, Instructor A: *"It [the activity] was more connected to their skills. It was because I was drawing more clear connections between [the ideas of] 'Here's my identity as a programmer' and 'Because I'm being asked very specific things about the software, I can actually use my expertise as a programmer."'*
>
> R1: *"Oh, so it sort of gives them that ability, if they are a little shaky in their [critical] identity to be like: 'No no, I can do this. I can talk about the software."'*
>
> Instructor A: *"Yeah, yeah, yeah."*

Similarly, Course B's TA described how many computing students were motivated by the desire to obtain a computing job or internship. They framed the kind of structured critical reflection involved in CIDER activities as something that could help students achieve those roles:

> Debrief+Post-Teaching interview, B-TA: *"I really want students to succeed, not just in class, but also academically. And I do think you can easily tie CIDER techniques to real life work. A lot of the common narrative [in CS spaces] is 'students go to industry,' right? And I think the reality is, in industry, you could easily apply CIDER thinking to how products are developed and project planning, [...] This can be nice and portable to go to different places with you."*

Whether by dismantling CS gatekeeping through approachable definitions, or leveraging students' prior CS knowledge to motivate critical interface design learning, instructors used these kinds of strategies to navigate student differences in their computing courses.

6.4.6 Theme 6: Making content applicable

Challenge 6: Cultivating agency and hope

General instructional challenge: *Students may question how course material applies to their broader lives or intended careers, and may not be motivated to engage without these connections.* Finally, students in many learning contexts question how course content might apply to their own future careers and broader lives. When motivating course content that connected technical topics to broader social narratives, Instructor A shared that they try to help students see how their perspectives might contribute unique insights in future careers:

> Pre-Teaching interview, Instructor A: *"If you want to do any kind of social argumentation, what value do you as a computer scientist bring? You have that ability to make that bridge between the [technical] details of the thing you're building, and the social questions you yourself or your co-workers are asking. And I think that is a unique place where, as a computer scientist [or] a software engineer you uniquely bring that, compared to, say, a product manager or a UX designers who may not be thinking as deeply about the technical details."*

Helping students make these kinds of connections can promote motivation to learn and engage with content, promoting agency and the development of discipline-specific identity. General instructional challenges in this vein were identified across all three courses, observed in data from all three main instructors and both sets of TAs.

Critical-specific nuance: *Critical activities that deal with weighty topics like oppression and bias can leave students with feelings of hopelessness; Particular care is needed to help students develop agency, hope, and habitual critical mindsets.*
Initial exposures to tough topics like bias and systematic oppression during critical instruction might leave students feeling sad or even "horrified" at the existence and scope of these problems. When discussing the results of their CIDER activity that critiqued the assumptions embedded in an overview of GDPR data privacy rights, Instructor B mentioned that it was difficult to get some

students past the initial shock over how poorly their data was typically handled, which prevented them from deeply engaging in the CIDER activity:

> Debriefing interview, Instructor B: *"There's also part of this, like, I'm interested in [getting students to critique] the specific details, and the students are all just horrified that this exists. [...] they're mostly freshmen, [they] haven't taken a class, or taken a step back on this type of issue. Maybe half of them are like, 'yeah, I already know this', but like about half of them are just like 'I've definitely not thought of this before, this is so horrifying and creepy.' So I feel like if I was going to get in more detail, I need to have gone through the first step with most of the students to get them past the 'Oh, no'."*

Getting students past the initial shock and negative emotions associated with exposure to critical topics is a key part of enabling them to begin figuring out how to tackle these issues in their own practice. Data from three main instructors and both sets of TAs indicated the presence of this critical-specific nuance around how to inspire agency and hope.

Mitigation strategies: *Instilling agency through small acts of resistance; Choosing optimism over pessimism; Shifting from single activities to ongoing critical mindsets.*
To help instill a sense of agency in the face of overwhelming systemic problems, Instructor A framed the CIDER activities as helping students practice small acts of resistance that could contribute to broader changes:

> Post-Teaching interview, Instructor A: *"One of the biggest things I think about now is, 'how can you make this more of an optimistic course?' In terms of not just stopping at critique, but also what we do about it. And I think that action component is not just an optimism for optimism's sake thing — I think it is a very direct way to give students power in the scenario when they might otherwise feel powerless. Sometimes it is especially true for students who are in this generation where you see so much stuff, there's all kind of terrible things happening the world [...] Giving them tools to address that in their own small ways, in ways that actually seem doable, and that are*

> *not asking the world from them, are actually immensely valuable. So I feel like CIDER kind of provides [the idea that] we're not talking about massive structural change. We're talking about things that are in your control. But they're things that if you speak about them, and you show that these are valuable things, that you can gradually shift to those broader systemic changes that are happening."*

To resist pessimistic narratives, Instructor B slightly modified the format of one of their CIDER activities to focus on the potential harms and benefits of social media for different users, rather than exclusionary assumptions. This activity, they hoped, could help students resist hopelessness:

> Post-Teaching interview, Instructor B: *"I really wanted to give them a chance to break out of the negative framing of social media that I see commonly. [...] But then I really wanted to break away from that, and also think about positives [of social media] as well, and not just 'social media is bad, but we're addicted to it.' [...] I think the universal pessimism throws out the good, denies the good. I want you to come away being able to build upon the good, and minimize or get rid of the bad. And if it's all bad, then there's nothing to do besides get rid of social media entirely, and that isn't going to work, and it's also not accurate."*

Instructor C stressed the importance of practicing critical computing interface design skills not only to develop proficiency, but also as a means of cultivating ongoing critical mindsets that students could carry with them into their broader lives:

> Debriefing interview, Instructor C: *"I think that next time I do this [activity] I would say something along the lines of, 'We have this [CIDER] protocol so that it becomes a habit.' So that it becomes a skill that you have, so that you're like constantly doing this. So it kind of runs like a background daemon. [...] It's a mindset. It's a skill to develop a mindset. So we're doing it explicitly, slowly, talking about each one, make you write down each one. So that it becomes a background habit."*

Though it may take extra care and effort on the part of instructors to create space for optimistic and hopeful responses to difficult critical topics, it appears important to ensuring students develop agency to take action toward dismantling intimidating systemic issues in their everyday lives and their future careers.

6.5 Discussion & Concluding Remarks

6.5.1 Summary of key results

In this study, I asked the following research questions:

- RQ1: What **instructional challenges** arise as computing educators integrate critical interface design topics into computing courses?
- RQ2: What **mitigation strategies** do computing educators employ to address the above challenges?

I investigated these questions through a 22-week Action Research community exploration of the pedagogical content knowledge (PCK) three computing educators and their TAs developed as they integrated a critical computing interface design method called CIDER into their courses. These community members all had existing commitments to integrating critical design activities into computing education. Through qualitative analysis of data from interviews, observations, and Slack messages with the instructors and TAs, I identified eleven salient instructional challenges that might arise when teaching and learning critical computing interface design concepts. Each instructional challenge had a *general instruction* component that might arise in any learning context, and a *critical-specific* nuance that seemed unique to efforts to teach critical interface design topics in computing courses. I organized descriptions of these challenges according to six overarching themes of instructional activity: preparing to teach a new topic; designing curricula; developing students' design skills; conducting interactive activities; accounting for student differences; and making content applicable to students. For each challenge, I also provided rich descriptions of one or more *mitigation strategies* that the instructors and TAs in the Action Research community used to address each of the eleven instructional challenges, contributing 23 mitigation strategies total. These descriptions of instructional challenges and mitigation strategies together comprise PCK foundations for teaching critical computing interface design.

6.5.2 Limitations

Although the insights I collected during the investigation were rich, some aspects of my methodology limit interpretations of my findings. Action Research studies do not attempt to control the environments under study, preferring instead to focus on authentically representing the experiences of community members during the timeframe of study. As a result, I cannot be sure exactly which aspects of the intervention and instructional support may have influenced the prevalence of different instructional challenges, nor can I claim that this list of instructional challenges and mitigations is exhaustive. Instructors and TAs involved in the study likely varied in their abilities to reflect upon and identify salient aspects of their own pedagogy, which may have influenced the data I captured to represent instructors' experiences in each course. To manage the scope of the study, I opted to collect data only from instructors and TAs and not from students themselves, so I cannot say how students may have experienced these courses or the CIDER activities themselves. Further, the situated nature of this study affords us deep insights into a specific community's engagement with a problem, but these results are not necessarily immediately generalizable to other computing educational contexts. I adhered to best practices for Action Research by safeguarding against methodological limitations through use of triangulation across several diverse learning contexts, but investigations at different types of learning institutions with different communities of educators and researchers teaching different students may surface different, complementary insights about the teaching and learning of critical computing interface design topics.

Further, several limitations arise from the institutional context of the university where I conducted this investigation. The community I engaged with was situated at a well-resourced research university with a strong design culture among its computing-centric departments, a noted focus on accessible technology research, and an extant community of computing educators and computing education researchers dedicated to improving CS education. Educators at this institution had the freedom to design their courses and integrate critical topics as they saw fit. I took this unique opportunity to bring together a community of people with already-existing commitments to integrating critical topics into computing courses, affording us insights that may not have been possible to garner elsewhere. However, not all educators in post-secondary computing departments have these favorable foundations to build upon. For instance, under the United States political climate at the time of writing, some states and institutions ban educators from even mentioning criticality or critical perspectives at all, precluding research into ideas such as the ones I explored here. The impacts of administrative and political constraints on instruction would have been much more prevalent had I attempted to conduct this investigation with communities in those kinds of learning environments. Similarly, I might expect different instructional challenges to arise in computing departments less favorable toward interface design topics in general, or in those that do not value excellence in teaching.

Finally, I acknowledge the positionalities of the Action Research community as researchers and educators with regards to the course design, activity integration, study design, data collection, and data analysis decisions. Given the expertise and interest of members in the Action Research community, I opted to explore criticality in computing education contexts through the lens of an inclusive interface design method as a focal intervention. This is far from the only conceptualization of criticality in computing; In fact, it is not even the most popular or widespread one [35,212]. While I hope that these insights can compliment and build upon existing work on criticality in computing education, results almost certainly would have differed if I chose another critical lens for the investigation. Further, each member of the Action Research community who participated in this study had pre-established interests in exploring best practices for developing critical computing knowledge and commitments to creating equitable and just computing education spaces. While this motivated us to deeply and rigorously engage with the investigation, my results should also be interpreted in light of these commitments. My interpretations of the findings, the data I chose to collect, and even the research questions themselves may have been different if the community comprised of different people with their own perspectives, backgrounds, context knowledge, and lived experiences. Finally, the analysts' interpretations of the qualitative analyses are influenced by our individual points of view from which we engaged with the data. There are particular perspectives and experiences that our analysts did not have access to based on their privileged positions within academia and within society as a whole. To truly enable equitable and just computing futures, we will need to expand non-extractive efforts to re-design computing spaces *with*, not *for* [20] stakeholders from many diverse types of marginalized populations, creating learning experiences that recognize and celebrate kinds of knowledge that are traditionally delegitimized under formalized education [278].

6.5.3 Implications & Future Work

Despite these limitations, my findings suggest a number of intriguing implications for critical HCI and computing education practice. The PCK foundations I identified in this study can inform pedagogical approaches and materials, especially those that highlight the potential power imbalances inherent in computing interface design processes. Another recurring insight noted by the instructors across contexts was the fact that integrating critical interface design topics into their course required more effort, and different kinds of effort, than non-critical topics might have. Some of this effort required instructors to reflect upon their own identities and positionalities with respect to the material (e.g. Challenge 1). Other times, this effort required instructors to actively deconstruct and resist hegemonic norms of existing computing education traditions (e.g. Challenges 3C, 5, and 6).

This work is very difficult to do without community and institutional support. Prior work has already noted that computing educators identify lack of institutional support as a major barrier to integrating basic accessibility topics into computing courses [165]; One might imagine that the support for integrating critical topics into computing is even lower. If we are to ensure that future computing professionals can critically reflect upon the implications of their interface design decisions, we will need to intentionally build these support structures. Future practice-oriented efforts in this space might bring together larger communities of educators with commitments to teaching critical computing interface design topics with a goal of developing instructional proficiency or teaching materials. Though there already exist multiple educator communities around teaching criticality in computing, the efforts to build these kinds of communities in HCI education are only beginning. A promising space to explore these possibilities might be the quickly-growing EduCHI [200] community of educators, researchers, and practitioners. While this group tends to focus on general computing interface design education, each year multiple papers describing pedagogical efforts around inclusive, ethical, or critical interface design are published at the venue.

My results also present several implications for critical computing & HCI education research. As mentioned in the Method section and in the above limitations, the Action Research community involved in this study was based at an institution with an overall favorable attitude toward critical interface design topics, with several researchers and educators at that institution exploring new ways to integrate criticality into computing spaces. Even with that favorable foundation, I *still* noted several instructional challenges that involved breaking down hegemonic norms of the computing field (as noted in the above paragraph), as well as indications that CS students were problem-solving in ways antithetical to deep engagement with critical problems (e.g. Challenges 3A and 4C). Systematic and concerted efforts to change the nature of computing education may be necessary to create learning contexts that effectively train students in critical and ethical reflection. Future research in this vein might attempt to identify exactly which aspects of a learning intuition make it more or less conducive to critical instruction. Researchers might also work alongside educators to identify needed resources and develop new pedagogical methods for teaching critical topics in computing education, iterating on interventions to develop theoretically-grounded best practices.

Chapter 7
DISCUSSION & CONCLUDING REMARKS

In this dissertation, I described four studies I conducted to explore different facets of the integration of inclusive design and computing education. This effort was in service of the broader goal of equipping technology creators with the skills they need to make less biased, more accessible, and more inclusive software and hardware interfaces, so that everyone can access the benefits of technology provides. To scope my operationalization of inclusive design, I specifically focused on pedagogical challenges and instructional strategies that supported computing students' abilities to make more inclusive *design decisions*, noting that the implicit and explicit assumptions held by designers can be a vector for computing interface design biases and exclusion.

In this chapter, I reinterpret my findings in light of broader implications for integrating inclusive design and computing education, discussing limitations and future work. Toward this goal, I reiterate my thesis statement here:

> Two forms of design decisions manifest within computing education: *program-space* design decisions that rely on technical implementation knowledge, and *problem-space* design decisions that rely on knowledge of the broader world. Computing students particularly struggle to learn problem-space design concepts, inhibiting their abilities to make inclusive design decisions. Explicit guidance on identifying the *assumptions about users* embedded in computing interfaces encourages students to make inclusive design decisions. When teaching inclusive computing interface design topics, computing instructors develop critical-specific pedagogical content knowledge (PCK) that differs from PCK for general computing instruction.

Finally, I conclude with some thoughts on how we might re-imagine the priorities of computing and human-computer interaction (HCI) education.

7.1 Understanding how computing education leverages design to help students situate technology within the world

In Chapter 3, I describe a study that investigated how design activity manifested within computing education contexts. Design is a distinct discipline with its own practices, tools, professions, and areas of scholarship. However, practitioners from other fields often leverage aspects of design in their own work, leading to subfields like engineering design and architecture design that are neither wholly design nor wholly the intersecting discipline. Similarly, design and computing are known to intersect in educational contexts [261]. At the time of this study, we did not yet have a clear characterization of the kinds of design activity that accompany computing topics, presenting challenges to teaching and learning these skills. This gap is particularly prevalent in K-12 computing education, where design is often used to promote student engagement but rarely studied as its own disciplinary phenomenon.

To better understand the nature and role of design in computing education, I led two qualitative, exploratory analyses of how design skills manifested in popular K-12 computing education curricula and activities. I found evidence to suggest two types of design activity within existing computing education curricula and standards:

- **Problem-space design: The "what" and the "why".** This kind of design answers questions like "What should this technology do (or enable users to do) within the broader world?" and "Why does the world need this technology?" Problem-space design manifested in learning objectives and activities about generating high-level requirements, understanding stakeholder perspectives, evaluating a product's fit against a set of real-world constraints, and justifying design rationale to peers or instructors.

- **Program-space design: The "how".** This kind of design answers the question "How should the available tools and resources be used to effectively implement this technology?" Given a set of problem-space requirements that define what the technology should do, program-space design involves choosing algorithms, data structures, function calls, and other coding constructs to meet those requirements. Program-space design decisions are often implemented by writing and executing code.

I found that these two types of computing design activity could exist independently, but they often overlapped, creating an intriguing intersection of discipline-specific computing design educational activity. Notably, several of the materials I analyzed implied both problem-space and program-space design practices within the same computing learning activities, entangling the "what" and "why" with the "how" of technology creation.

This study contributed a dual-type model of design activity in computing education, laying foundations for further research into problem-space design pedagogy in computing education contexts. While effective program-space design decision-making leverages students' technical implementation knowledge, effective problem-space design decision-making leverages students' knowledge of the broader world and of potential users' needs. To make *inclusive* design decisions that take into account the broad spectrum of human diversity, computing students need to develop competency in problem-space design. While problem-space design can be considered *non-disciplinary* design, since these activities might be found in any design process regardless of medium, program-space design can be considered *disciplinary* design within computing education, since the nuances of program-space design tend to be unique to the medium of code and software creation. Both kinds of activities are arguably important for a thorough understanding of technology design, but the skillsets required to conduct them are distinct. At the time of the study, there was some existing research into pedagogy for program-space design skills in computing education (c.f. efforts to support algorithmic thinking, pseudocode writing, or debugging strategies), but there was comparatively little that focused on teaching and learning program-space design skills.

These results substantiate the first claim in this dissertation's thesis statement by providing evidence for the dual-type model of the different kinds of design decisions that may manifest in computing education contexts, as well as highlighting the role of problem-space design skills in creating technology that fits well within the world. This study also noted that several of the computing education activities which asked students to make problem-space design decisions required them to draw upon knowledge about the world. As noted in Chapter 2, technology designers who solely rely on their knowledge of the world without explicitly attending to the needs of different user groups tend to make less usable, less inclusive computing interfaces. The problem-space and program-space model of design provides theoretical foundations for understanding pedagogical practices around helping computing students learn to make more inclusive computing interface design decisions.

7.2 Identifying computing students' learning challenges in introductory interface design courses

In Chapter 4, I describe a study that investigated computing students' learning challenges in post-secondary human-computer interaction (HCI) design courses. Understanding student learning challenges is generally considered a key component of effective pedagogy [108]. I conducted two qualitative studies consisting of surveys and interviews with (1) computing students enrolled in introductory interface design courses and (2) educators who teach interface design to computing students. Qualitative analysis of their responses revealed 18 types of learning challenges students might experience in computing interface design education, including difficulties around the mechanics of design work, project management skills, the wicked nature of design problems, and distorted perspectives involving design. If these kinds of learning challenges are not adequately addressed, computing students may find it difficult to do any kind of design work at all, much less make *inclusive* design decisions.

This study contributed a set of learning challenges that arise in introductory computing interface design education, with descriptions of each one from students and educators who had experienced or observed them. When mapped onto the problem-space and program-space model of design activity within computing education, these learning challenges represent issues largely with problem-space design learning, though some of sit in the intersection of both kinds of design (see Figure 4.1 at the beginning of the chapter). If these kinds of learning challenges are not addressed within introductory design education, computing students may find it difficult to do any kind of design work at all, much less make *inclusive* design decisions when they are creating computing interfaces.

The last category of student learning challenges around distorted perspectives on design are particularly important for this pursuit. If a computing student devalues the impact of their design decisions (`RUSH`) or doesn't understand why inclusion matters (`WARP`), they will likely fall back on common, normative, or even exclusionary design patterns (`STUCK`) without reflecting upon their own biases (`BIAS`) or whether those design patterns actually support different kinds of interaction styles well (`DIVRS`). Without these kinds of understandings, it can be difficult for computing students to see themselves as designers (`ID`) and thus as responsible for the design decisions they make when creating technology. These results substantiate the second claim in this dissertation's thesis statement: that computing students struggle to learn problem-space design concepts, which may inhibit their abilities to make inclusive design decisions. Interventions to teach inclusive design skills to computing students should take these learning challenges into account.

7.3 Developing and evaluating a pedagogical technique to teach inclusive interface design skills

In Chapter 5, I describe a study that investigated how to help computing students learn to recognize computing interface design bias and practice making inclusive design decisions. I created the created the theoretically-grounded CIDER assumption elicitation technique, an educational analytical design evaluation method. CIDER (which stands for Critique, Imagine, Design, Expand, Repeat) provides explicit guidance on identifying embedded *assumptions about users* present in the design of computing interfaces. This technique leverages guided critique, brainstorming, and peer feedback to help students understand how normative assumptions can manifest in biased design features and exclude people from interacting with a design as intended. CIDER uses a strategy called *assumption elicitation* to help computing students learn to identify embedded assumptions about users within computing interfaces, understand how those assumptions might lead to exclusion, and redesign the interfaces to be more inclusive. The technique was created to specifically target several of the student learning challenges described in Chapter 4, specifically those around distorted perspectives on design work which might inhibit inclusive design learning.

I conducted a mixed-method case study evaluating CIDER's efficacy in a post-secondary computing interface design course with 40 undergraduate computing students. Through a series of individual and collaborative CIDER activities, we found that use of the technique contributed to statistically significant increases in students' self-efficacy for inclusive design activity. Qualitative analysis of the assumptions students identified in CIDER activities indicated that the technique helped them expand their understandings of how technological interfaces might exclude different minoritized populations more and more over time. Several students mentioned that CIDER activities had prompted "aha" moments by revealing the multitude of ways a design might be biased and how different assumptions might exclude different user groups. In follow-up interviews several weeks later, some students even reported that the CIDER activities had long-lasting impacts, encouraging them to value inclusion, consider computing interfaces more critically, consider diverse groups of users, and make inclusive design decisions in their subsequent design work.

These results substantiate the third claim in this dissertation's thesis statement: That providing explicit guidance on identifying the assumptions about users embedded in computing interfaces encourages students to make inclusive design decisions. Future work on integrating inclusive design topics into computing educations could more deeply explore the framing of assumptions about users that seemed to be effective in this study, teasing apart exactly what about this framing seems to help students gain new design understandings.

7.4 Surfacing pedagogical content knowledge for teaching inclusive interface design topics in computing courses

In Chapter 6, I describe a Action Research study that investigated the pedagogical content knowledge (PCK) computing educators developed as they integrated critical interface design topics into their computing courses. The instructors used the CIDER assumption elicitation technique from Chapter 5 as their focal pedagogical method, with support from a small team of computing education researchers with expertise in critical computing education (including myself). This community coalesced around a shared desire to understand how best to help students critically reflect upon the impacts and implications of computing interface design decisions.

Over the course of 22 weeks (two academic terms), I led a series of semi-structured interviews, co-designed learning materials, observed courses, and facilitated a Slack community with three computing instructors and their teaching assistants (TAs) for three different courses. To scope the investigation, I elected to focus on a specific element of PCK: the *instructional challenges* educators encountered and the *mitigation strategies* they employed to address these challenges. Qualitative analysis of data from the above sources surfaced 11 types of instructional challenges, each of which had a *general instruction* component that might manifest in any computing or design classroom, and a *critical-specific* nuance that was unique to the teaching and learning of critical interface design topics within computing courses. I also described 23 different mitigation strategies instructors used to address these challenges, with accompanying indications of efficacy where available.

Instructors spoke about how they used CIDER activities as opportunities to help students reflect more broadly on the impacts of technological design decisions on society, showcasing actionable ways that students could make technology more inclusive. Notably, the instructors who participated in the study leveraged the actionability of CIDER's assumption elicitation approach to help students regain hope and agency after disheartening initial exposures to systemic problems of design bias. These insights substantiate the claim that the critical-specific (PCK) computing instructors develop as they teach critical and inclusive computing interface design differs from PCK for general computing instruction. The mitigation strategies used by instructors in this study might also be used to help integration other critical topics into computing education, supporting broader goals of more critical computing education [178].

7.5 Contributions to Computing and HCI Education Research

The work I present in this dissertation is largely formative, motivated largely by a lack of HCI education research and computing design education research to build upon. However, I also contribute foundational understandings of the role inclusive design activity plays within critical computing education and pedagogical bases for more rigorous HCI pedagogy.

7.5.1 Contributions to computing education

There have been several recent calls for more critical computing education [178, 313], including efforts to describe the landscape of critical computing education interventions [212] and analyses of ethics-centered rhetoric in the computing education research field [35]. The work presented in this dissertation builds upon justice-centered approaches to computing education [195, 212] which aim to help computing students critically reflect upon the individual and societal impacts of the technologies they create. Through these four projects, I showed that integrating inclusive interface design ideas into computing education is an effective way of helping students reason through the roles and responsibilities they have in perpetuating normative design biases and upholding technological hegemony.

Computing education research rarely engages with design as a theoretical grounding for pedagogy. Recent work suggests that engaging more deeply with design might improve the rigor and quality of work in the computing education field [261]. Based on the insights gleaned from my work, I argue that framing critical and ethical computing problems in terms of design is actually crucial for supporting new understandings of how technological biases contribute to marginalization. Design-based pedagogical interventions, where computing students create and reflect upon their design decisions, may provide a means of reifying potentially abstract ideas of justice, inclusion, and ethical decision-making.

Further, my results suggest that teaching and learning computing through a design-informed, assumption-based lens can help computing students better understand the subjectivity inherent to technology creation. Notions of technological objectivity (e.g., the belief that algorithms cannot be biased because they are "just math") are often perpetuated in dominant computing narratives [18], and thus also in many computing education structures as well [287]. As referenced throughout this dissertation, ignoring the differential impacts technology has upon different user groups leads to marginalization and minoritization. Highlighting the influence and impact of implicit assumptions in the technology creation process resists these narratives, introducing ideas of subjectivity and, as a result, potential for bias. This, in turn, creates fruitful conditions for students to begin critically reflecting on the ethics and impacts of technology. The interventions I describe in this dissertation are one way to introduce computing students to inclusive design and other critical perspectives on technology, which can hopefully inform future efforts in the space.

7.5.2 Contributions to HCI education

HCI education research is a relatively young field, generally lacking deep pedagogical foundations unique to the discipline [302]. To remedy this, HCI educators and researchers tend to draw upon applied design education research in related fields (e.g. engineering design education, STEM design education). However, as discussed in the related work in Chapter 3, there has been very little work that investigates the specific intersection of *computing* and design education unique to HCI [231]. The work I present in this dissertation contributes some of the first theoretical foundations for rigorous inclusive computing interface design pedagogy, and by extension, for critical HCI education research.

One known difficult problem within HCI education is how best to help students develop a sense of responsibility for their design decisions and agency to make meaningful positive impact in their future careers [54,125,295]. Results from this dissertation work indicate that teaching HCI students to consider computing interfaces through the lens of *assumptions about users* can help them situate their creations within the broader world. This framing reveals the disproportionately large individual and societal impacts that can arise from something as small as an ill-considered design decision, and as a result, exemplifies aspects of the power designers hold over user experience. Based on my findings, assumption-based HCI education appears to facilitate students' understandings of HCI *threshold concepts*—topics that, once understood, irrevocably change a person's perceptions of a discipline and their role within it [172]. For instance, in Chapter 5, some students relayed that engaging in assumption-based interface design activities led them to rethink their approach not only to design work, but to many more aspects of their lives as well, like consumer purchasing habits or how they structured their living spaces. Assumption-based inclusive design activities provide a concrete way to promote critical and inclusive design agency.

My results also indicate that the challenges computing students and educators face when teaching and learning inclusive and critical interface design topics are meaningfully different than those that arise in other disciplinary design education contexts. Accordingly, we will also need unique pedagogical methods, educator supports, and resources that address these differences. Some recent work has laid foundations for critical computing educator training programs [175]. The mitigation strategies described in Chapter 6 could build upon this work to inform further efforts specific to computing interface design, contributing initial best practices for critical and inclusive HCI education.

7.6 Limitations & Opportunities for Future Work

While the work presented in this dissertation contributed several novel insights to advance computing and HCI education, there remain some limitations to our results due to knowledge, time, or feasibility constraints. In the following section, I discuss some of the most relevant limitations, framing each as opportunities for future work on integrating inclusive design and computing education.

7.6.1 Evaluating the quality of students' redesign proposals

My work to this point has mostly focused on *introducing* computing students to inclusive interface design ideas. The studies presented in Chapters 4 and 5 were motivated partially by a need to understand how computing students react when initially exposed to unfamiliar interface design concepts. Due to the state of the literature at the time these studies were conducted, these foundations were necessary to explore and catalog before diving deeper. However, we will need to understand other aspects of students' inclusive design learning processes to facilitate rigorous instruction.

One salient limitation of the CIDER technique based work (Chapters 5 and 6) is that I did not investigate anything about the *quality* of redesign proposals that students generated during the DESIGN stage of CIDER activities (i.e. the changes they would make to an artifact's design to make it more inclusive). Evaluating the quality of design ideas is a known difficult problem in adjacent domains like engineering design education [119]. I noted in Chapter 5 that though I collected over 2000 redesign proposal ideas from students in the original CIDER technique case study, I did not yet have a basis for evaluating the quality of these design proposals, so I opted not to analyze that data. Further, in Chapter 6, instructors in the Action Research community asked several times for advice on evaluating the quality of students' responses to CIDER activities (including their redesign proposals) for the purposes of grading. Most instructors opted to simply grade on participation, but doing so provides little formative feedback to students about how they can improve their application of inclusive design knowledge.

Future work in this space can explore which aspects of students' redesign proposals might signify high-quality responses. For instance, what kinds of changes to designs are students proposing on these inclusive design activities, and are those changes actually making interfaces more inclusive? Are the proposed changes merely surface level, or do they meaningfully change the form or function of the computing interface? What kinds of redesign proposals signal deep understandings of inclusive interface design concepts, and which might indicate misconceptions that should be addressed? Insights from exploring metrics for quality of inclusive redesigns could benefit both researchers and practitioners.

7.6.2 Supporting students' reasoning around design tradeoffs

My overarching goal with this dissertation was to contribute actionable pedagogical strategies for integrating introductory inclusive interface design topics into computing education. Many of my insights revolve around how to effectively *complicate* computing students' understandings of interface design, enabling them to transition from viewing design as "inessential", "easy", or "commonsense" [55, 91] to a complicated process that merits careful consideration to ensure they are effectively addressing the needs of people from different marginalized groups. However, a limitation of this work is that I do not deeply investigate how to help computing students reason through design tradeoffs and *resolve* the aforementioned complications. Tradeoffs are especially important to consider within the inclusive interface design space because a feature that increases access for one group might make a technology less accessible to another.

Choosing between alternatives (EVAL) is one of the student learning difficulties I identified in my exploration of HCI learning challenges (Chapter 4). Some work in engineering design education literature has explored how tradeoffs can be used as a signifier of students' design knowledge, but the issue has received little-to-no attention within computing design education or HCI education literature. Future work in this space could investigate methods to support computing students' analytical reasoning around design tradeoffs. According to the results presented in Chapter 6, these interventions will likely need to be mindful of the ways that they encourage hope, agency, and optimism in students, instead of leaving them feeling like they cannot ever make a "correct" design decision in the face of systemic design bias and technological marginalization.

7.6.3 Leveraging design traditions beyond inclusive design

I grounded this dissertation in the approach of *inclusive design*, conceptualizing a critical justice-centered definition of inclusion in Chapter 2 to more strongly connect the approach to issues of power and systemic marginalization. While inclusive design approaches do typically emphasize recognizing and designing for the needs of marginalized users [5, 210], there are some limitations to this framing.

For instance, traditional conceptions of inclusive design operationalize the notion of inclusion as a proportion of a target population that can successfully use or benefit from a design as intended. A design is considered "more inclusive" when the ratio of users in the targeted population who can successfully use the artifact is as close to one as possible. Keates et al.'s *Inclusive Design Cube (IDC)* model has three axes, based on varying demands placed on the sensory, cognitive, or motor capacity of a potential user, with the volume of the cube representing what proportion of a target population has those capabilities. Designs which account for larger volumes (i.e. those that support broader spectra of ability) are considered more inclusive under the IDC model [168]. In order to quantify included and excluded populations, Keates and colleagues leverage survey data from disability surveys conducted with British adults, which was collected in such a way to allow them to assign proportions of population to locations on the cube's three axes of disability severity [167]. Recently, Goodman-Deane et al. extended the ideas behind the IDC model to apply to user characteristics such as technology access, psychological factors, and access to support, noting the difficulty inherent in mapping some of these characteristics to ordinal scales [121].

Making the number of people excluded salient in definitions of inclusion *can* help make abstract notions of inclusion and exclusion more concrete, directly refuting those who view design as a purely aesthetic pursuit (an unfortunately common misconception in technical fields [91,193,229]). Quantifiable metrics such as population coverage may also help designers and developers advocate for inclusive design to clients, especially if the proportion excluded is large enough to impact potential adoption and sales [318]. However, this focus runs the risk of erasing groups that are small in number, yet disproportionately impacted by software exclusion—groups which tend to include multiply marginalized individuals with intersectionally disadvantaged identities. This can lead to these groups being excluded so long as the number impacted falls within some nominally "acceptable" amount of marginal exclusion. Further, conceptions of inclusive design that measure it in terms of population proportion rely on data sets to contain accurate information about potentially excluded individuals in the first place [121]. While it may be possible to collect this data about some aspects of inclusion, it becomes harder to find accurate estimates for others attributes of inclusion, such as those who hold marginalized ability-based, gender, racial, ethnic, sexual, cultural, or class identities, largely due to the ways that typical classification structures encourage compliance to oppressive identity norms [25].

Another limitation of using inclusive design as a basis for this work is that it can unintentionally perpetuate technosolutionism. The CIDER technique presented and evaluated in Chapters 5 and 6 focuses on helping students recognize exclusionary design assumptions within a computing interface and practice proposing changes that could improve inclusion for different populations. I chose to focus on the skill of making relatively small changes to existing designs in order to build students' inclusive design self-efficacy through concrete actions that seemed achievable at their introductory proficiency levels with design concepts. However, in the words of Instructor A of the Action Research community from Chapter 6, *"Making an unjust technology more inclusive is not necessarily better!"* The inclusive design approach by itself does not directly engage with issues of systemic biases, hegemonic power imbalances, or intentionally exploitative design patterns. This broader framing and discussion is left to educators to provide for their students, though (as discussed throughout this document) HCI and computing educators often lack content knowledge in specifically these areas.

Future work in this area should explore how grounding computing interface design pedagogy in different theoretical traditions might prompt students (and educators) to develop different understandings of design decisions and assumptions as vectors for bias and marginalization. Several design approaches emphasizing different aspects of user experience might provide fruitful starting points, such as value-sensitive design [99, 100] or ability-based design [221, 307]. Design justice approaches [70] might be particularly interesting theoretical traditions in which to ground HCI and computing interface design pedagogy, since they are community-centric and explicitly emphasize the resistance of power imbalances inherent to traditional design processes[1].

Future work should also explore how to integrate notions of *critical refusal* [58, 105, 106] into computing students' technology design education. As noted by Course A's TAs in Chapter 6, some students already feel that learning about justice, ethics, and inclusion in computing is pointless, since *"we can talk about solving problems all day here in our ivory academia tower. But if [they] go into [their] industry internship and they're just like: 'Code some stuff for the Defense Department' or whatever, [they] don't feel like [they] can say anything about it."* As prominent decision-makers in the design of future technology, computing students *need* to feel empowered to speak up against technological injustice and be aware that they can choose to say "no" to implementing technology that runs counter to their values. Effective justice-centered HCI and computing design education can become one avenue of building this confidence.

7.6.4 Developing and evaluating inclusive design interventions across educational contexts

My initial interest in this work was pragmatically motivated by experiences and observations of difficulties introducing inclusion topics in design-antagonistic computer science departments where students had little background knowledge of HCI or accessibility. However, I actually conducted most of this formative work within design-favorable informatics and computer science departments

[1]If I were to re-do the many years of this dissertation work with the knowledge I now have, I would probably choose to situate this work in design justice traditions from the start.

at an institution where students tend to encounter HCI and accessibility concepts several times over the course of their education. I believe the design-favorable environment was indeed necessary for the initial development of the ideas presented in this dissertation, especially for expanding my own conceptions of inclusive and justice-centered approaches to technology design. The climate of the institution also ensured existence of a large enough group of critically-minded computing education researchers and educators for me to leverage participatory, community-based methods to knowledge production, like Chapter 6's Action Research investigation. However, this institutional background also presents a salient limitation of this work in the tensions between its motivation and the contexts in which it was implemented.

To develop cross-cutting understandings of HCI and computing interface design pedagogy, the insights presented in this dissertation should be implemented and evaluated in different educational contexts. Future work in this space can involve collaboration with computing educators to create computing interface design learning materials that fit their learning contexts, then evaluating the effectiveness of their adapted interventions. There are many types of computing educators of varied levels and with diverse learning goals that might be interested in integrating inclusive interface design topics into their courses, such as university faculty, primary and secondary school teachers, community workshop leaders, professional development program leaders, and bootcamp instructors. Each of these educational contexts would provide rich insights into what computing interface design teaching and learning looks like with different audiences, contributing deeper foundations for future computing and HCI education work to build upon.

7.6.5 Training educators to teach critical computing interface design principles

As mentioned throughout this dissertation, most graduate education programs lack formal pedagogical training, leaving computing faculty to develop much of their pedagogy through experience in the form of pedagogical content knowledge [266]. Making PCK explicit (like I did in Chapter 6) can support higher quality instruction [67], but to have the opportunity to do so, it needs to be communicated in some way to the educators who will use it. As it stands, there are relatively few teacher education and outreach efforts focused on training computing design educators and HCI educators, much less on training them on teaching *inclusive* or *critical* computing interface design principles.

Future work on preparing HCI educators and computing educators to talk about critical and inclusive interface design concepts should focus on how best to communicate the tensions and nuances that accompany real-world problems of marginalization. For instance, my dissertation work contributed the CIDER pedagogical method to teach inclusive design skills, intending it to be applicable in many different educational contexts. However, a major limitation of this work is that it does not explicitly address the potential pitfall of students and/or educators developing a "checkbox mentality" around the method, seeing it as a panacea for "fixing" inclusion issues without consideration for the broader landscape of inclusive design activity. Educators who are not familiar with concepts of accessibility and inclusion might see CIDER activities as something they can simply "drop into" their curriculum for a requisite lesson on accessibility, without providing framing knowledge of systemic bias, marginalization, and tradeoffs that students need to develop nuanced understandings of computing interface design. Students who are introduced to the method in this way might erroneously develop a belief that they can simply do a CIDER activity to "solve" access and inclusion issues in their designs for good. This kind of instruction might lead to more harm than good as students transition to their careers without recognizing the limitations of their perspectives.

One way to address this problem might be through designing workshops for instructors and faculty that provide foundational knowledge about accessibility, inclusion, systemic bias, and design work. For CIDER, access to teaching materials and resources about the technique might only be provided after educators complete one of these training sessions. Justice-centered CS teacher education programs [175] can provide a starting model for these efforts, but they likely cannot be copied directly, since my results indicate that the knowledge needed to effectively teach inclusive interface design is meaningfully different than the knowledge required to teach general computing concepts. Further, to fit into educators' overloaded schedules, these training sessions will need to be relatively short, necessitating decisions about what topics to cover and how deeply. Results from this work would directly benefit practitioners by supporting the development of cohorts of critical and inclusive HCI educators, while also providing novel insights into HCI teacher education. The latter is a largely unexplored research area, with only my dissertation work and a few other extant prior works [228, 253] providing theoretical foundations.

7.6.6 Extending insights to new domains of computing interface design

My dissertation work focuses largely on *problem-space* (Chapter 3) inclusive design of computing interfaces, helping students learn to make interface design decisions that better reflect the needs of marginalized users. A wide-open space future work should explore is that of *program-space* inclusive design at the level of code structure and data models.

Software that operates on data about users typically contains some code-level representation of user identity (name, gender, race or ethnicity options, nationality, etc.), which is stored in data structures or databases. Software developers design data schemas to represent identity information and verification rules that determine what types of data are considered valid. The categorical rigidity imposed by these data schemas is an inherent property of code-level representation: Algorithms typically operate poorly on input unstructured, ambiguous input, so algorithm designers and developers attempt to impose some order by defining classification systems which segment different types of users based on pre-defined attributes. Bowker and Star define three properties of a classification system: That there are consistent, unique principles by which items are sorted; that the categories of system are mutually exclusive; and that the system is complete and covers all potential instances of what items are or can be [25]. Classification systems for user identities are baked into software at the level of data structures and re-inscribed by validation rules and interface features that restrict the form and type of information these fields accept [21]. Static, literal, rigidly-bounded schemes for storing representations of user identity function well only when a user's identify fits into the allowable bounds of the system. However, prior work also shows that these kinds of classifications often erase the presence of marginalized identities [18, 70], especially racial and ethnic identities, for which there is no apolitical classification scheme [284]. Further, identity is not static, and it is difficult for these systems to account for marginalized [131, 171] and changing [129] identities in a way that fully respects the humanity of the user being represented.

Conceiving of identity as intersectional [14, 245, 256] almost by definition breaks classification systems, in that identities can no longer be represented by a single (or a set of) mutually exclusive categories.

Because code-level design decisions can have such disruptive impacts on diverse users, we should work toward justice-centered software development practices to ensure that diverse users can fully participate in today's digital worlds, enabling developers to critically evaluate the impacts of code-level design decisions as much as their interface design decisions. Achieving this goal will require change in several areas. Computing education will to emphasize the impacts software development choices have on real-world users and impress a sense of responsibility for these decisions onto future developers. Policy changes to ensure equitable software access must be implemented, then enforced in practice. Organizations in the computing industry will need to train their employees to understand and instantiate justice-centered development principles in their processes, from the designers, project managers, and marketing teams who decide what software should do in the world, to the software architects and developers who decide how those requirements should be translated into code and data. All of these avenues present intriguing and important challenges for future work in program-space inclusive design research to address.

7.7 Concluding Remarks

To situate the conclusion of this work, consider this quote from Instructor C, one of the educators in our Action Research community from Chapter 6:

> Instructor C: *"I think the point of education is not to make a workforce. The point of education is to make good people. And when you argue, 'well, it's not to create the workforce', you must also say 'what is your alternative perspective of education?"'*

In this dissertation, I present an alternative perspective on computing and HCI education: One that helps future computing professionals critically reflect upon the impacts of their design decisions, considering the broad range of human diversity while doing so. Realizing this vision requires a fundamental re-imagining of the priorities of HCI and computing education. What might computing education contexts look like if they were focused not only on creating more technically adept developers, but instead on *making good people*? The work described in this dissertation provides insights into how we might achieve this and similar goals by integrating inclusive interface design concepts and methods into computing education contexts.

Of course, revolution cannot happen overnight; nor can it survive in isolation. While the work of systemic change may seem daunting, I encourage readers to reflect upon the advice our instructors from Chapter 6 shared for cultivating agency and hope in the face of large-scale, seemingly overwhelming problems. We cannot solve these issues as individuals, but only in community. We must push past the initial discomfort and create the educational spaces we wish to see through small acts of deliberate, optimistic, and thoughtful resistance. The need for computing students to develop critical design competencies increases by the day as technology becomes more and more integrated into society and everyday life. There is no better time to start this work than now.

BIBLIOGRAPHY

[1] Lucas F. Abreu, Glivia A.R. Barbosa, Ismael S. Silva, and Natalia S. Santos. Characterizing Software Requirements Elicitation Processes: A Systematic Literature Review. In *Proceedings of the XII Brazilian Symposium on Information Systems on Brazilian Symposium on Information Systems: Information Systems in the Cloud Computing Era - Volume 1*, SBSI 2016, pages 26:192–26:199, Porto Alegre, Brazil, Brazil, 2016. Brazilian Computer Society.

[2] Edith Ackermann. Piaget's Constructivism, Papert's Constructionism: What's the difference? 2001.

[3] Piotr D. Adamczyk and Michael B. Twidale. Supporting multidisciplinary collaboration: requirements from novel HCI education. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI'07)*, page 1073, San Jose, California, USA, 2007. ACM Press.

[4] Robin S. Adams, Jennifer Turns, and Cynthia J. Atman. Educating effective engineering designers: the role of reflective practice. *Design Studies*, 24(3):275–294, May 2003.

[5] Adobe Systems, Inc. Inclusive Design at Adobe, 2022.

[6] Christopher Alexander. Notes on the Synthesis of Form. *Harvard Graduate School of Design*, January 1964.

[7] Farshid Anvari, Deborah Richards, Michael Hitchens, and Hien Minh Thi Tran. Teaching user centered conceptual design using cross-cultural personas and peer reviews for a large cohort of students. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering Education and Training*, ICSE-SEET '19, pages 62–73, Montreal, Quebec, Canada, May 2019. IEEE Press.

[8] L. Bruce Archer. *Systematic Method for Designers.* Council of Industrial Design, 1965.

[9] Henrik Artman and Mattias Arvola. Studio life: The construction of digital design competence. *Nordic Journal of Digital Literacy*, 3(02):78–96, 2008.

[10] Owen Astrachan and Amy Briggs. The CS principles project. *ACM Inroads*, 3(2):38–42, 2012.

[11] Cynthia J. Atman, Robin S. Adams, Monica E. Cardella, Jennifer Turns, Susan Mosborg, and Jason Saleem. Engineering Design Processes: A Comparison of Students and Expert Practitioners. *Journal of Engineering Education*, 96(4):359–379, 2007.

[12] Cynthia J. Atman, Justin R. Chimka, Karen M. Bursic, and Heather L. Nachtmann. A comparison of freshman and senior engineering design processes. *Design Studies*, 20(2):131–152, March 1999.

[13] Albert Bandura. Self-efficacy. *The Corsini encyclopedia of psychology*, pages 1–3, 2010.

[14] Shaowen Bardzell. Feminist HCI: Taking Stock and Outlining an Agenda for Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1301–1310, New York, NY, USA, 2010. ACM.

[15] Lecia Jane Barker, Kathy Garvin-Doxas, and Michele Jackson. Defensive Climate in the Computer Science Classroom. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '02, pages 43–47, New York, NY, USA, 2002. ACM.

[16] R. J. Barnes, D. C. Gause, and E. C. Way. Teaching the Unknown and the Unknowable in Requirements Engineering Education. In *2008 Requirements Engineering Education and Training*, pages 30–37, September 2008.

[17] Eric P. S. Baumer and Jed R. Brubaker. Post-userism. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 6291–6303, Denver, Colorado, USA, May 2017. Association for Computing Machinery.

[18] Ruha Benjamin. Race After Technology: Abolitionist tools for the new Jim code. *Social Forces*, 2019.

[19] Cynthia L. Bennett, Burren Peil, and Daniela K. Rosner. Biographical Prototypes: Reimagining Recognition and Disability in Design. In *Proceedings of the 2019 on Designing Interactive Systems Conference*, DIS '19, pages 35–47, New York, NY, USA, June 2019. Association for Computing Machinery.

[20] Cynthia L. Bennett and Daniela K. Rosner. The Promise of Empathy: Design, Disability, and Knowing the "Other". In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–13, New York, NY, USA, May 2019. Association for Computing Machinery.

[21] Rena Bivens. The gender binary will not be deprogrammed: Ten years of coding gender on Facebook. *New Media & Society*, 19(6):880–898, June 2017. Publisher: SAGE Publications.

[22] Sara Black. Marx's Ghost in the Shell: Troubling Techno-Solutionism in Post-Secondary Education and Training Policy Imaginaries. *Education as Change*, 26:23 pages–23 pages, October 2022.

[23] Barry W. Boehm. A spiral model of software development and enhancement. *Computer*, (5):61–72, 1988.

[24] Aikaterini Bourazeri and Simone Stumpf. Co-designing smart home technology with people with dementia or Parkinson's disease. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction*, NordiCHI '18, pages 609–621, New York, NY, USA, September 2018. Association for Computing Machinery.

[25] Geoffrey C. Bowker and Susan Leigh Star. *Sorting Things Out: Classification and Its Consequences*. MIT Press, August 2000.

[26] Engin Bozdag. Bias in algorithmic filtering and personalization. *Ethics and information technology*, 15(3):209–227, 2013.

[27] Ofra Brandes and Michal Armoni. Using Action Research to Distill Research-Based Segments of Pedagogical Content Knowledge of K-12 Computer Science Teachers. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '19, pages 485–491, New York, NY, USA, 2019. ACM. event-place: Aberdeen, Scotland Uk.

[28] Carol B. Brandt, Katherine Cennamo, Sarah Douglas, Mitzi Vernon, Margarita McGrath, and Yolanda Reimer. A Theoretical Framework for the Studio as a Learning Environment. *International Journal of Technology and Design Education*, 23(2):329–348, 2013. Publisher: Springer.

[29] John D. Bransford, Ann L. Brown, and Rodney R. Cocking. *How People Learn*, volume 11. Washington, DC: National academy press, 2000.

[30] Andrea Branzi. We Are the Primitives. *Design Issues*, 3(1):23–27, 1986.

[31] Robin Braun, Wayne Brookes, Roger Hadgraft, and Zenon Chaczko. Assessment Design for Studio-Based Learning. In *Proceedings of the Twenty-First Australasian Computing Education Conference*, ACE '19, pages 106–111, New York, NY, USA, 2019. ACM.

[32] Samantha Breslin and Bimlesh Wadhwa. Exploring Nuanced Gender Perspectives Within the HCI Community. In *Proceedings of the India HCI 2014 Conference on Human Computer Interaction*, IndiaHCI '14, pages 45:45–45:54, New York, NY, USA, 2014. ACM.

[33] Samantha Breslin and Bimlesh Wadhwa. Towards a Gender HCI Curriculum. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 1091–1096, New York, NY, USA, 2015. ACM.

[34] David C. Brown. Assumptions in Design and in Design Rationale. page 5.

[35] Noelle Brown, Benjamin Xie, Ella Sarder, Casey Fiesler, and Eliane S. Wiese. Teaching Ethics in Computing: A Systematic Literature Review of ACM Computer Science Education Publications. *ACM Transactions on Computing Education*, November 2023. Just Accepted.

[36] Amy Bruckman, Maureen Biggers, Barbara Ericson, Tom McKlin, Jill Dimond, Betsy DiSalvo, Mike Hewner, Lijun Ni, and Sarita Yardi. "Georgia Computes!": Improving the Computing Education Pipeline. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, pages 86–90, New York, NY, USA, 2009. ACM.

[37] Emeline Brulé and Katta Spiel. Negotiating Gender and Disability Identities in Participatory Design. In *Proceedings of the 9th International Conference on Communities & Technologies - Transforming Communities*, C&T '19, pages 218–227, New York, NY, USA, June 2019. Association for Computing Machinery.

[38] Louis L. Bucciarelli. *Designing Engineers*. MIT Press, 1994.

[39] Richard Buchanan. Wicked Problems in Design Thinking. *Design Issues*, 8(2):5–21, 1992.

[40] Janet E. Burge. Design rationale: Researching under uncertainty. *AI EDAM*, 22(4):311–324, November 2008.

[41] Janet E. Burge and David C. Brown. Rationale-Based Support for Software Maintenance. In Allen H. Dutoit, Raymond McCall, Ivan Mistrík, and Barbara Paech, editors, *Rationale Management in Software Engineering*, pages 273–296. Springer, Berlin, Heidelberg, 2006.

[42] Sheryl Burgstahler. Universal Design: Implications for Computing Education. *Transactions on Computing Education (TOCE)*, 11(3):19:1–19:17, October 2011.

[43] Kenneth Burke. Four Master Tropes. *The Kenyon Review*, 3(4):421–438, 1941.

[44] Margaret Burnett, Anicia Peters, Charles Hill, and Noha Elarief. Finding gender-inclusiveness software issues with GenderMag: A field investigation. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, CHI '16, pages 2586–2598, New York, NY, USA, 2016. ACM.

[45] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. GenderMag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*, 28(6):760–787, 2016.

[46] Maria Camacho. David Kelley: From Design to Design Thinking at Stanford and IDEO. *She Ji: The Journal of Design, Economics, and Innovation*, 2(1):88–101, 2016.

[47] Adam R. Carberry, Hee-Sun Lee, and Matthew W. Ohland. Measuring engineering design self-efficacy. *Journal of Engineering Education*, 99(1):71–79, 2010.

[48] John M. Carroll and Judith Reitman Olson. Chapter 2 - Mental Models in Human-Computer Interaction. In MARTIN Helander, editor, *Handbook of Human-Computer Interaction*, pages 45–65. North-Holland, Amsterdam, January 1988.

[49] John M. Carroll and Mary Beth Rosson. Getting Around the Task-artifact Cycle: How to Make Claims and Design by Scenario. *ACM Trans. Inf. Syst.*, 10(2):181–212, April 1992.

[50] Sharon M. Carver, Richard Lehrer, Tim Connell, and Julie Erickson. Learning by hypermedia design: Issues of assessment and implementation. *Educational Psychologist*, 27(3):385–404, 1992.

[51] LaVar J. Charleston. A qualitative investigation of African Americans' decision to pursue computing science degrees: Implications for cultivating career choice and aspiration. *Journal of Diversity in Higher Education*, 5(4):222–243, 2012.

[52] Alain J. F. Chiaradia, Louie Sieh, and Frances Plimmer. Values in urban design: A design studio teaching approach. *Design Studies*, 49:66–100, March 2017.

[53] Parmit K. Chilana, Rishabh Singh, and Philip J. Guo. Understanding Conversational Programmers: A Perspective from the Software Industry. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16)*, pages 1462–1472, Santa Clara, California, USA, 2016. ACM Press.

[54] Shruthi Sai Chivukula, Aiza Hasib, Ziqing Li, Jingle Chen, and Colin M. Gray. Identity Claims that Underlie Ethical Awareness and Action. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, number 295, pages 1–13. Association for Computing Machinery, New York, NY, USA, May 2021.

[55] Elizabeth F. Churchill, Anne Bowser, and Jennifer Preece. Teaching and Learning Human-computer Interaction: Past, Present, and Future. *interactions*, 20(2):44–53, March 2013.

[56] Elizabeth F. Churchill, Anne Bowser, and Jennifer Preece. The future of HCI education: a flexible, global, living curriculum. *Interactions*, 23(2):70–73, February 2016.

[57] Marika Cifor and Patricia Garcia. Gendered by Design: A Duoethnographic Study of Personal Fitness Tracking Systems. *ACM Transactions on Social Computing*, 2(4):15:1–15:22, January 2020.

[58] Marika Cifor, Patricia Garcia, TL Cowan, Jasmine Rault, Tonia Sutherland, Anita Chan, Jennifer Rode, Anna Lauren Hoffmann, Niloufar Salehi, and Lisa Nakamura. Feminist data manifest-no. *Cit. on*, 119, 2019.

[59] Code.org. CS Discoveries Curriculum Guide 2018 - 2019, 2018. Retrieved January 2019.

[60] Code.org. The State of K-12 Computer Science. Technical report, Code.org, 2019.

[61] College Board. AP Computer Science Principles: The Course, February 2016.

[62] College Board. AP Computer Science A: The Course, 2019.

[63] Patricia Hill Collins. *Black feminist thought: Knowledge, consciousness, and the politics of empowerment*. Routledge, 2002.

[64] Computer Science Teachers Association (2017). CSTA K-12 Computer Science Standards, Revised 2017, 2017.

[65] Sunny Consolvo, Beverly Harrison, Ian Smith, Mike Y. Chen, Katherine Everitt, Jon Froehlich, and James A. Landay. Conducting In Situ Evaluations for and With Ubiquitous Computing Technologies. *International Journal of Human–Computer Interaction*, 22(1-2):103–118, April 2007.

[66] Sarah Cooney. Riding the Bus in Los Angeles: Creating Cultural Micro-Exposures via Technology. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, number 13, pages 1–9. Association for Computing Machinery, New York, NY, USA, May 2021.

[67] Rebecca Cooper, John Loughran, and Amanda Berry. Science Teachers' PCK. *Berry, A., Friedrichsen, P. & Loughran, J., Re-examining Pedagogical Content Knowledge in Science Education*, pages 60–74, 2015.

[68] Steve Cooper, Shuchi Grover, Mark Guzdial, and Beth Simon. A Future for Computing Education Research. *Communications of the ACM*, 57(11):34–36, October 2014.

[69] Sasha Costanza-Chock. Design Justice, A.I., and Escape from the Matrix of Domination. *Journal of Design and Science*, July 2018.

[70] Sasha Costanza-Chock. *Design Justice: Community-led practices to build the worlds we need.* MIT Press, 2020.

[71] Kimberle Crenshaw. Mapping the margins: Intersectionality, identity politics, and violence against women of color. *Stan. L. Rev.*, 43:1241, 1990. Publisher: HeinOnline.

[72] John W. Creswell. *Research design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications, Thousand Oaks, CA, US, 3rd edition, 2008.

[73] David P. Crismond and Robin S. Adams. The Informed Design Teaching and Learning Matrix. *Journal of Engineering Education*, 101(4):738–797, 2012.

[74] Nigel Cross. Designerly ways of knowing. *Design Studies*, 3(4):221–227, October 1982.

[75] Alma Leora Culén. HCI Education: Innovation, Creativity and Design Thinking. *International Conferences on Advances in Computer-Human Interactions*, pages 125–130, 2015.

[76] Ethan Danahy, Eric Wang, Jay Brockman, Adam Carberry, Ben Shapiro, and Chris B. Rogers. LEGO-based Robotics in Higher Education: 15 Years of Student Creativity. *International Journal of Advanced Robotic Systems*, 11(2):27, February 2014.

[77] James Davis, Michael Lachney, Zoe Zatz, William Babbitt, and Ron Eglash. A Cultural Computing Curriculum. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 1171–1175, New York, NY, USA, 2019. ACM.

[78] Elise Deitrick, Brian T. O'Connell, and R. Benjamin Shapiro. The Discourse of Creative Problem Solving in Childhood Engineering Education. 2014.

[79] Elise Deitrick, R. Benjamin Shapiro, Matthew P. Ahrens, Rebecca Fiebrink, Paul D. Lehrman, and Saad Farooq. Using Distributed Cognition Theory to Analyze Collaborative Computer Science Learning. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, pages 51–60, New York, NY, USA, 2015. ACM.

[80] Halime Demirkan and Yasemin Afacan. Assessing creativity in design education: Analysis of creativity factors in the first-year design studio. *Design Studies*, 33(3):262–278, May 2012.

[81] Design Justice Network. Read the Principles, 2018. https://designjustice.org/read-the-principles.

[82] Dan Ding, Rory A. Cooper, and Jon Pearlman. Incorporating Participatory Action Design into Research and Education. page 6, 2007.

[83] Carl DiSalvo. *Adversarial Design*. The MIT Press, 2012.

[84] Kees Dorst and Nigel Cross. Creativity in the design process: co-evolution of problem-solution. *Design Studies*, 22(5):425–437, September 2001.

[85] Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L. Schwartz, and Scott R. Klemmer. Parallel Prototyping Leads to Better Design Results, More Divergence, and Increased Self-efficacy. *ACM Trans. Comput.-Hum. Interact.*, 17(4):18:1–18:24, December 2010.

[86] Hugh Dubberly. *How do you design? A Compendium of Models*. January 2008.

[87] Wendy DuBow and Allison-Scott Pruitt. NCWIT Scorecard: The Status of Women in Computing [2019 Update], 2019.

[88] Caitlin Duncan and Tim Bell. A Pilot Computer Science and Programming Course for Primary School Students. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, WiPSCE '15, pages 39–48, New York, NY, USA, 2015. ACM.

[89] Ruth Dunn. Minority Studies, 2021.

[90] Anthony Dunne. *Hertzian Tales: Electronic Products, Aesthetic Experience, and Critical Design*. The MIT Press, 2006.

[91] Alistair D. N. Edwards, Peter Wright, and Helen Petrie. HCI education: We are failing - why? In *In Proceedings of HCI Educators Workshop 2006*, pages 23–24, 2006.

[92] Jayne Everson, F. Megumi Kivuva, and Amy J. Ko. "A Key to Reducing Inequities in Like, AI, is by Reducing Inequities Everywhere First": Emerging Critical Consciousness in a Co-Constructed Secondary CS Classroom. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*, volume 1 of *SIGCSE 2022*, pages 209–215, New York, NY, USA, February 2022. Association for Computing Machinery.

[93] Exploring Computer Science. ECS Curriculum: Requests & Downloads, June 2015. Retrieved June 26, 2019.

[94] Anthony Faiola. The design enterprise: Rethinking the HCI education paradigm. *Design Issues*, 23(3):30–45, 2007.

[95] Juan-Miguel Fernandez-Balboa and Jim Stiehl. The generic nature of pedagogical content knowledge among college professors. *Teaching and teacher education*, 11:293–306, 1995.

[96] Casey Fiesler, Natalie Garrett, and Nathan Beard. What Do We Teach When We Teach Tech Ethics? A Syllabi Analysis. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, pages 289–295, New York, NY, USA, February 2020. Association for Computing Machinery.

[97] Allan Fisher and Jane Margolis. Unlocking the Clubhouse: The Carnegie Mellon Experience. *SIGCSE Bulletin*, 34(2):79–83, June 2002.

[98] Paulo Freire. Pedagogy of the oppressed. In *Toward a Sociology of Education*, pages 374–386. Routledge, 2020.

[99] Batya Friedman. Value-sensitive Design. *interactions*, 3(6):16–23, December 1996.

[100] Batya Friedman and David Hendry. *Value Sensitive Design: Shaping Technology with Moral Imagination*. MIT Press, May 2019.

[101] Batya Friedman and Helen Nissenbaum. Bias in computer systems. *ACM Transactions on Information Systems*, 14(3):330–347, July 1996.

[102] Carol Frieze, Jeria L. Quesenberry, Elizabeth Kemp, and Anthony Velázquez. Diversity or Difference? New Research Supports the Case for a Cultural Perspective on Women in Computing. *Journal of Science Education and Technology*, 21(4):423–439, August 2012.

[103] Jeffrey E. Froyd, Phillip C. Wankat, and Karl A. Smith. Five major shifts in 100 years of engineering education. *Proceedings of the IEEE*, 100(Special Centennial Issue):1344–1360, 2012.

[104] Daniel D. Garcia, Valerie Barr, Mark Guzdial, and David J. Malan. Rediscovering the Passion, Beauty, Joy, and Awe: Making Computing Fun Again, Part 6. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 379–380, New York, NY, USA, 2013. ACM.

[105] Patricia Garcia, Tonia Sutherland, Marika Cifor, Anita Say Chan, Lauren Klein, Catherine D'Ignazio, and Niloufar Salehi. No: Critical Refusal as Feminist Data Practice. In *Companion Publication of the 2020 Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '20 Companion, pages 199–202, New York, NY, USA, October 2020. Association for Computing Machinery.

[106] Patricia Garcia, Tonia Sutherland, Niloufar Salehi, Marika Cifor, and Anubha Singh. No! Re-imagining Data Practices Through the Lens of Critical Refusal. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW2):315:1–315:20, November 2022.

[107] Andrea Alessandro Gasparini. Perspective and Use of Empathy in Design Thinking. page 6, 2015.

[108] Julie Gess-Newsome. Pedagogical Content Knowledge: An Introduction and Orientation. In Julie Gess-Newsome and Norman G. Lederman, editors, *Examining Pedagogical Content Knowledge: The Construct and its Implications for Science Education*, Science & Technology Education Library, pages 3–17. Springer Netherlands, Dordrecht, 1999.

[109] Guiseppe Getto and Fred Beecher. Toward a model of UX education: Training UX designers within the academy. *IEEE Transactions on Professional Communication*, 59(2):153–164, 2016.

[110] Maliheh Ghajargar and Jeffrey Bardzell. Synthesizing Opposites: Technical Rationality and Pragmatism in Design. *The Design Journal*, 22(sup1):2031–2044, April 2019.

[111] Sucheta Ghoshal and Sayamindu Dasgupta. Design Values in Action: Toward a Theory of Value Dilution. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*, DIS '23, pages 2347–2361, New York, NY, USA, July 2023. Association for Computing Machinery.

[112] Susannah Go and Brian Dorn. Thanks for Sharing: CS Pedagogical Content Knowledge Sharing in Online Environments. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE '16*, pages 27–36, M&#252;nster, Germany, 2016. ACM Press.

[113] Ashok K. Goel, Swaroop Vattam, Bryan Wiltgen, and Michael Helms. Cognitive, collaborative, conceptual and creative — Four characteristics of the next generation of knowledge-based CAD systems: A study in biologically inspired design. *Computer-Aided Design*, 44(10):879–900, October 2012.

[114] Vinod Goel and Peter Pirolli. The structure of design problem spaces. *Cognitive Science*, 16(3):395–429, July 1992.

[115] J. A. Goguen and C. Linde. Techniques for requirements elicitation. In *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 152–164, January 1993.

[116] Göran Goldkuhl. Pragmatism vs interpretivism in qualitative information systems research. *European Journal of Information Systems*, 21(2):135–146, March 2012. Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/ejis.2011.54.

[117] Gabriela Goldschmidt and Paul A. Rodgers. The design thinking approaches of three different groups of designers based on self-reports. *Design Studies*, 34(4):454–471, July 2013.

[118] Gabriela Goldschmidt and Anat Litan Sever. Inspiring design ideas with texts. *Design Studies*, 32(2):139–155, March 2011.

[119] Molly Hathaway Goldstein, Camilo Vieira Mejia, Robin Adams, Şenay Purzer, and Mitch Zielinski. Developing a measure of quality for engineering design artifacts. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–7, October 2016.

[120] Joanna Goode and Gail Chapman. Exploring Computer Science. Technical report, Computer Science Equity Alliance, 2011.

[121] J. Goodman-Deane, M. Bradley, S. Waller, and P. J. Clarkson. Quantifying Exclusion for Digital Products and Interfaces. In Patrick Langdon, Jonathan Lazar, Ann Heylighen, and Hua Dong, editors, *Designing for Inclusion*, pages 140–149. Springer International Publishing, Cham, 2020.

[122] Sukeshini Grandhi. Educating Ourselves on HCI Education. *interactions*, 22(6):69–71, October 2015.

[123] Colin Gray, Seda Yilmaz, Shanna Daly, Colleen Seifert, and Richard Gonzalez. Idea Generation Through Empathy: Reimagining the 'Cognitive Walkthrough'. *2015 ASEE Annual Conference and Exposition*, June 2015.

[124] Colin M. Gray. Evolution of design competence in UX practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1645–1654. ACM, 2014.

[125] Colin M. Gray, Austin L. Toombs, and Shad Gross. Flow of competence in UX design practice. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3285–3294. ACM, 2015.

[126] Tom Gross. Human-Computer Interaction Education and Diversity. In Masaaki Kurosu, editor, *Human-Computer Interaction. Theories, Methods, and Tools*, Lecture Notes in Computer Science, pages 187–198. Springer International Publishing, 2014.

[127] Mark Guzdial. Balancing Teaching CS Efficiently with Motivating Students. *Communications of the ACM*, 60(6):10–11, May 2017.

[128] Jan H. van Driel, Nico Verloop, and Wobbe de Vos. Developing science teachers' pedagogical content knowledge. *Journal of Research in Science Teaching - J RES SCI TEACH*, 35:673–695, 1998.

[129] Oliver L. Haimson, Jed R. Brubaker, Lynn Dombrowski, and Gillian R. Hayes. Digital Footprints and Changing Networks During Online Identity Transitions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2895–2907, San Jose, California, USA, May 2016. Association for Computing Machinery.

[130] Rich Halstead-Nussloch and Han Reichgelt. Teaching HCI in a "Crowded" Computing Curriculum. *J. Comput. Sci. Coll.*, 29(2):184–190, December 2013.

[131] Foad Hamidi, Morgan Klaus Scheuerman, and Stacy M. Branham. Gender Recognition or Gender Reductionism? The Social Implications of Embedded Gender Recognition Systems. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 1–13, Montreal QC, Canada, April 2018. Association for Computing Machinery.

[132] David Hammer and Leema K. Berland. Confusing claims for data: A critique of common practices for presenting qualitative research on learning. *Journal of the Learning Sciences*, 23(1):37–46, 2014.

[133] Martyn Hammersley and Paul Atkinson. *Ethnography: Principles in practice*. Routledge, 2007.

[134] Steve Harrison and Deborah Tatar. On Methods. *Interactions*, 18(2):10–11, March 2011.

[135] Rex Hartson. Cognitive, physical, sensory, and functional affordances in interaction design. *Behaviour & Information Technology*, 22(5):315–338, September 2003.

[136] Gillian R. Hayes. Knowing by doing: Action Research as an approach to HCI. In *Ways of Knowing in HCI*, pages 49–68. Springer, 2014.

[137] Heather C. Hill, Deborah Loewenberg Ball, and Stephen Schilling. Unpacking pedagogical content knowledge: Conceptualizing and measuring teachers' topic-specific knowledge of students. *Journal for research in mathematics education*, 39:372–400, 2008.

[138] Heather C. Hill, Brian Rowan, and Deborah Loewenberg Ball. Effects of teachers' mathematical knowledge for teaching on student achievement. *American Educational Research Journal*, 42(2):371–406, 2005.

[139] Cindy E. Hmelo, Douglas L. Holton, and Janet L. Kolodner. Designing to Learn About Complex Systems. *Journal of the Learning Sciences*, 9(3):247–298, July 2000.

[140] Cindy E. Hmelo-Silver. Problem-Based Learning: What and How Do Students Learn? *Educational Psychology Review*, 16(3):235–266, September 2004.

[141] Cindy E. Hmelo-Silver and Merav Green Pfeffer. Comparing expert and novice understanding of a complex system from the perspective of structures, behaviors, and functions. *Cognitive science*, 28(1):127–138, 2004.

[142] Beryl Hoffman, Ralph Morelli, and Jennifer Rosato. Student Engagement is Key to Broadening Participation in CS. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 1123–1129, New York, NY, USA, 2019. ACM.

[143] bell hooks. *Teaching To Transgress: Education as the practice of freedom*. Routledge, 1994.

[144] Chenglie Hu. Can Students Design Software?: The Answer Is More Complex Than You Think. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 199–204, New York, NY, USA, 2016. ACM.

[145] Peter Hubwieser, Marc Berges, Johannes Magenheim, Niclas Schaper, Kathrin Bröker, Melanie Margaritis, Sigrid Schubert, and Laura Ohrndorf. Pedagogical Content Knowledge for Computer Science in German Teacher Education Curricula. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, WiPSE '13, pages 95–103, New York, NY, USA, 2013. ACM. event-place: Aarhus, Denmark.

[146] Peter Hubwieser, Johannes Magenheim, Andreas Mühling, and Alexander Ruf. Towards a conceptualization of pedagogical content knowledge for computer science. In *Proceedings of the ninth annual international ACM conference on international computing education research*, ICER '13, pages 1–8, New York, NY, USA, 2013. ACM.

[147] C. D. Hundhausen, D. Fairbrother, and M. Petre. An Empirical Study of the "Prototype Walkthrough": A Studio-Based Activity for HCI Education. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19(4):26:1–26:36, December 2012.

[148] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. Direct Manipulation Interfaces. *Human–Computer Interaction*, 1(4):311–338, December 1985. Publisher: Taylor & Francis _eprint: https://doi.org/10.1207/s15327051hci0104_2.

[149] Hilary Hutchinson, Wendy Mackay, Bo Westerlund, Benjamin B. Bederson, Allison Druin, Catherine Plaisant, Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, Heiko Hansen, Nicolas Roussel, and Björn Eiderbäck. Technology Probes: Inspiring Design for

and with Families. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 17–24, New York, NY, USA, 2003. ACM.

[150] N. H. Ibrahim, J. Surif, A. H. Abdullah, and N. A. S. Sabtu. Comparison of pedagogical content knowledge between expert and novice lecturers in teaching and learning process. In *2014 International Conference on Teaching and Learning in Computing and Engineering*, pages 240–246, April 2014.

[151] Hengameh Irandoust. The Logic of Critique. *Argumentation*, 20(2):133–148, October 2006.

[152] Terry Irwin. Transition Design: A Proposal for a New Area of Design Practice, Study, and Research. *Design and Culture*, 7(2):229–246, April 2015.

[153] David G. Jansson and Steven M. Smith. Design Fixation. *Design Studies*, 12(1):3–11, 1991. Publisher: Elsevier.

[154] Karl K. Jeffries. Diagnosing the creativity of designers: individual feedback within mass higher education. *Design Studies*, 28(5):485–497, September 2007.

[155] Jennifer Leigh Brown. Empathy Mapping: A Guide to Getting Inside a User's Head | UX Booth, June 2018.

[156] Lin Jia, Yasmine N. Elglaly, Catherine M. Baker, and Kristen Shinohara. Infusing Accessibility into Programming Courses. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, number 231, pages 1–6. Association for Computing Machinery, New York, NY, USA, May 2021.

[157] Bobbie Johnson. Privacy no longer a social norm, says Facebook founder. *The Guardian*, January 2010.

[158] John Christopher Jones. *Design methods: seeds of human futures*. Wiley-Interscience, 1970.

[159] Yasmin B. Kafai. Playing and Making Games for Learning: Instructionist and Constructionist Perspectives for Game Studies. *Games and Culture*, 1(1):36–40, January 2006.

[160] Yasmin B. Kafai. *Minds in Play : Computer Game Design As A Context for Children's Learning*. Routledge, December 2012.

[161] Yasmin B. Kafai, Eunkyoung Lee, Kristin Searle, Deborah Fields, Eliot Kaplan, and Debora Lui. A Crafts-Oriented Approach to Computing in High School: Introducing Computational Concepts, Practices, and Perspectives with Electronic Textiles. *Transactions on Computing Education (TOCE)*, 14(1):1:1–1:20, March 2014.

[162] Y.B. Kafai, M.L. Franke, C.C. Ching, and J.C. Shih. Game Design as an Interactive Learning Environment for Fostering Students' and Teachers' Mathematical Inquiry. *International Journal of Computers for Mathematical Learning*, 3(2):149–184, May 1998.

[163] Yvonne Kao, Katie D'Silva, Aleata Hubbard, Joseph Green, and Kimkinyona Cully. Applying the Mathematical Work of Teaching Framework to Develop a Computer Science Pedagogical Content Knowledge Assessment. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, pages 888–893, New York, NY, USA, 2018. ACM. event-place: Baltimore, Maryland, USA.

[164] Vibha Kaushik and Christine A. Walsh. Pragmatism as a Research Paradigm and Its Implications for Social Work Research. *Social Sciences*, 8(9):255, September 2019. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

[165] Saba Kawas, Laura Vonessen, and Amy J. Ko. Teaching Accessibility: A Design Exploration of Faculty Professional Development at Scale. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 983–989, New York, NY, USA, February 2019. Association for Computing Machinery.

[166] Simeon Keates. Pragmatic research issues confronting HCI practitioners when designing for universal access. *Universal Access in the Information Society*, 5(3):269–278, November 2006.

[167] Simeon Keates and P. John Clarkson. Countering design exclusion through inclusive design. In *Proceedings of the 2003 conference on Universal usability*, CUU '03, pages 69–76, New York, NY, USA, June 2002. Association for Computing Machinery.

[168] Simeon Keates, P. John Clarkson, Lee-Anne Harrison, and Peter Robinson. Towards a practical inclusive design approach. In *Proceedings on the 2000 conference on Universal Usability*, CUU '00, pages 45–52, New York, NY, USA, November 2000. Association for Computing Machinery.

[169] Cazembe Kennedy and Eileen T. Kraemer. What Are They Thinking?: Eliciting Student Reasoning About Troublesome Concepts in Introductory Computer Science. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, Koli Calling '18, pages 7:1–7:10, New York, NY, USA, 2018. ACM. event-place: Koli, Finland.

[170] Gyorgy Kepes. Education of Vision. 1965.

[171] Os Keyes. The Misgendering Machines: Trans/HCI Implications of Automatic Gender Recognition. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):88:1–88:22, November 2018.

[172] Ahmed Kharrufa and Colin Gray. Threshold Concepts in HCI Education. In *2nd Annual ACM SIGCHI Symposium on HCI Education (EduCHI 2020)*, 2020.

[173] Dimitris Kiritsis. Closed-loop PLM for intelligent products in the era of the Internet of things. *Computer-Aided Design*, 43(5):479–501, May 2011.

[174] Amy J. Ko. A three-year participant observation of software startup software evolution. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track*, pages 3–12. IEEE Press, 2017.

[175] Amy J. Ko, Anne Beitlers, Jayne Everson, Brett Wortzman, and Dan Gallagher. Proposing, Planning, and Teaching an Equity- and Justice-Centered Secondary Pre-Service CS Teacher Education Program. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, pages 583–589, New York, NY, USA, March 2023. Association for Computing Machinery.

[176] Amy J. Ko and Parmit K. Chilana. Design, discussion, and dissent in open bug reports. In *Proceedings of the 2011 iConference*, pages 106–113. ACM, 2011.

[177] Amy J. Ko and Richard E. Ladner. AccessComputing promotes teaching accessibility. *ACM Inroads*, 7(4):65–68, 2016.

[178] Amy J. Ko, Alannah Oleson, Mara Kirdani-Ryan, Yim Register, Benjamin Xie, Mina Tari, Matthew Davidson, Stefania Druga, and Dastyni Loksa. It is time for more critical CS education. *Communications of the ACM*, 63(11):31–33, October 2020.

[179] Matthew J. Koehler and Richard Lehrer. Designing a hypermedia tool for learning about children's mathematical cognition. *Journal of Educational Computing Research*, 18(2):123–145, 1998.

[180] Ron Kohavi, Alex Deng, Brian Frasca, Toby Walker, Ya Xu, and Nils Pohlmann. Online Controlled Experiments at Large Scale. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1168–1176, New York, NY, USA, 2013. ACM.

[181] Janet L. Kolodner, Paul J. Camp, David Crismond, Barbara Fasse, Jackie Gray, Jennifer Holbrook, Sadhana Puntambekar, and Mike Ryan. Problem-Based Learning Meets Case-Based Reasoning in the Middle-School Science Classroom: Putting Learning by Design(tm) Into Practice. *Journal of the Learning Sciences*, 12(4):495–547, October 2003.

[182] Janet L. Kolodner, David Crismond, Jackie Gray, Jennifer Holbrook, and Sadhana Puntambekar. Learning by design from theory to practice. In *Proceedings of the international conference of the learning sciences*, volume 98, pages 16–22, 1998.

[183] Chinmay E. Kulkarni, Richard Socher, Michael S. Bernstein, and Scott R. Klemmer. Scaling Short-answer Grading by Combining Peer Assessment with Algorithmic Scoring. In *Proceedings of the First ACM Conference on Learning @ Scale Conference*, L@S '14, pages 99–108, New York, NY, USA, 2014. ACM.

[184] P. Lago and H. van Vliet. Explicit assumptions enrich architectural models. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 206–214, May 2005.

[185] Aubrey Lawson, Eileen T. Kraemer, S. Megan Che, and Cazembe Kennedy. A Multi-Level Study of Undergraduate Computer Science Reasoning About Concurrency. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '19, pages 210–216, New York, NY, USA, 2019. ACM. event-place: Aberdeen, Scotland Uk.

[186] Bryan Lawson. Cognitive Strategies in Architectural Design. *ERGONOMICS*, 22:59–68, January 1979.

[187] Jintae Lee and Kum-Yew Lai. What's in Design Rationale? *Human–Computer Interaction*, 6(3-4):251–280, September 1991.

[188] Catherine Legg and Christopher Hookway. Pragmatism. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2021 edition, 2021.

[189] Meir Manny Lehman and Juan Fernándezl-Ramil. The role and impact of assumptions in software engineering and its products. In *Rationale Management in Software Engineering*, pages 313–328. Springer, 2006.

[190] Richard Lehrer. Authors of knowledge: Patterns of hypermedia design. In *Computers as cognitive tools*, pages 205–236. Routledge, 2013.

[191] Gay Lemons, Adam Carberry, Chris Swan, Linda Jarvin, and Chris Rogers. The benefits of model building in teaching engineering design. *Design Studies*, 31(3):288–309, May 2010.

[192] Sarah Lewthwaite and David Sloan. Exploring Pedagogical Culture for Accessibility Education in Computing Science. In *Proceedings of the 13th Web for All Conference*, W4A '16, pages 3:1–3:4, New York, NY, USA, 2016. ACM.

[193] Paul Luo Li, Amy J. Ko, and Andrew Begel. Cross-disciplinary perspectives on collaborations with software engineers. In *Cooperative and Human Aspects of Software Engineering (CHASE), 2017 IEEE/ACM 10th International Workshop on*, pages 2–8. IEEE, 2017.

[194] Neomi Liberman, Yifat Ben-David Kolikant, and Catriel Beeri. In-service Teachers Learning of a New Paradigm: A Case Study. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop*, ICER '09, pages 43–50, New York, NY, USA, 2009. ACM. event-place: Berkeley, CA, USA.

[195] Kevin Lin. CS Education for the Socially-Just Worlds We Need: The Case for Justice-Centered Approaches to CS in Higher Education. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*, volume 1 of *SIGCSE 2022*, pages 265–271, New York, NY, USA, February 2022. Association for Computing Machinery.

[196] Min Liu. The effect of hypermedia authoring on elementary school students' creative thinking. *Journal of Educational Computing Research*, 19(1):27–51, 1998.

[197] Min Liu. Enhancing learners' cognitive skills through multimedia design. *Interactive Learning Environments*, 11(1):23–39, 2003. Publisher: Taylor & Francis.

[198] Stephanie Ludi. Introducing Accessibility Requirements through External Stakeholder Utilization in an Undergraduate Requirements Engineering Course. In *29th International Conference on Software Engineering (ICSE'07)*, pages 736–743, May 2007. ISSN: 1558-1225.

[199] Craig M. MacDonald and Michael E. Atwood. Changing perspectives on evaluation in HCI: past, present, and future. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 1969–1978, New York, NY, USA, April 2013. Association for Computing Machinery.

[200] Craig M. MacDonald, Olivier St-Cyr, Colin M. Gray, Leigh Ellen Potter, Carine Lallemand, Anna Vasilchenko, Jaisie Sin, Anna R. L. Carter, Caroline Pitt, Eunice Sari, Deepak Ranjan Padhi, and Ajit G. Pillai. EduCHI 2022: 4th Annual Symposium on HCI Education. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI EA '22, pages 1–5, New York, NY, USA, April 2022. Association for Computing Machinery.

[201] Murni Mahmud, Idyawati Hussein, Abu Osman Md Tap, and Nor Laila Md Noor. HCI Knowledge - Missing in Practice? In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, OzCHI '13, pages 511–514, New York, NY, USA, 2013. ACM.

[202] N. Mangano, T. D. LaToza, M. Petre, and A. van der Hoek. How Software Designers Interact with Sketches at the Whiteboard. *IEEE Transactions on Software Engineering*, 41(2):135–156, February 2015.

[203] M. M. Mantei. An HCI Continuing Education Curriculum for Industry. *SIGCHI Bull.*, 20(3):16–18, January 1989.

[204] Martin N. Marger. *Race and ethnic relations: American and global perspectives*. Cengage Learning, 2014.

[205] Katherine McCoy. Information and Persuasion: Rivals or Partners? *Design Issues*, 16(3):80–83, September 2000.

[206] D. Scott McCrickard, C. M. Chewar, and Jacob Somervell. Design, science, and engineering topics?: teaching HCI with a unified method. In *Proceedings of the 35th SIGCSE technical symposium on computer science education*, SIGCSE '04, pages 31–35, New York, NY, USA, 2004. ACM.

[207] Tom McKlin, Taneisha Lee, Dana Wanzer, Brian Magerko, Doug Edwards, Sabrina Grossman, Emily Bryans, and Jason Freeman. Accounting for Pedagogical Content Knowledge in a Theory of Change Analysis. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, pages 157–165, New York, NY, USA, 2019. ACM. event-place: Toronto ON, Canada.

[208] Christopher Mendez. *The InclusiveMag Method: A Start Towards More Inclusive Software for Diverse Populations*. PhD thesis, Oregon State University, 2020. Publisher: Oregon State University.

[209] Danaë Metaxa-Kakavouli, Kelly Wang, James A. Landay, and Jeff Hancock. Gender-Inclusive Design: Sense of Belonging and Bias in Web Interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 1–6, New York, NY, USA, April 2018. Association for Computing Machinery.

[210] Microsoft. Inclusive Design, 2022.

[211] Punyashloke Mishra and Matthew J. Koehler. Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge. 2006.

[212] Luis Morales-Navarro and Yasmin B. Kafai. Conceptualizing Approaches to Critical Computing Education: Inquiry, Design, and Reimagination. In Mikko Apiola, Sonsoles López-Pernas, and Mohammed Saqr, editors, *Past, Present and Future of Computing Education Research : A Global Perspective*, pages 521–538. Springer International Publishing, Cham, 2023.

[213] Michael J. Muller and Sarah Kuhn. Participatory Design. *Commun. ACM*, 36(6):24–28, June 1993.

[214] Mitchell J. Nathan and Anthony Petrosino. Expert Blind Spot Among Preservice Teachers. *American Educational Research Journal*, 40(4):905–928, January 2003.

[215] National Academies of Sciences, Engineering, and Medicine. *How people learn II: Learners, contexts, and cultures*. National Academies Press, 2018.

[216] Eduardo Navas, Owen Gallagher, and Borrough, xtine. *The Routledge companion to Remix studies*. Routledge, 2014.

[217] Timothy Neate, Aikaterini Bourazeri, Abi Roper, Simone Stumpf, and Stephanie Wilson. Co-Created Personas: Engaging and Empowering Users with Diverse Needs Within the Design Process. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–12, New York, NY, USA, May 2019. Association for Computing Machinery.

[218] George Nelson. *Problems of Design*. Whitney Library of Design, 1965.

[219] Maria Adriana Neroni and Nathan Crilly. Whose ideas are most fixating, your own or other people's? The effect of idea agency on subsequent design behaviour. *Design Studies*, 60:180–212, January 2019.

[220] Jakob Nielsen and Rolf Molich. Heuristic Evaluation of User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pages 249–256, New York, NY, USA, 1990. ACM.

[221] Amelie Nolte, Jacob O. Wobbrock, Torben Volkmann, and Nicole Jochems. Implementing Ability-Based Design: A Systematic Approach to Conceptual User Modeling. *ACM Transactions on Accessible Computing*, July 2022. Just Accepted.

[222] Donald A. Norman. *Emotional Design: Why We Love (or Hate) Everyday Things*. Basic Books, 2004. Google-Books-ID: z2jvRlqhdlwC.

[223] Donald A. Norman and Pieter Jan Stappers. DesignX: Complex Sociotechnical Systems. *She Ji: The Journal of Design, Economics, and Innovation*, 1(2):83–106, December 2015.

[224] Ihudiya Finda Ogbonnaya-Ogburu, Angela D.R. Smith, Alexandra To, and Kentaro Toyama. Critical Race Theory for HCI. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–16, New York, NY, USA, April 2020. Association for Computing Machinery.

[225] Laura Ohrndorf and Sigrid Schubert. Measurement of Pedagogical Content Knowledge: Students' Knowledge and Conceptions. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, WiPSE '13, pages 104–107, New York, NY, USA, 2013. ACM. event-place: Aarhus, Denmark.

[226] Alannah Oleson. CIDER: A Method to Teach Practical Critical Software Design Skills. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2*, ICER '22, pages 7–9, New York, NY, USA, August 2022. Association for Computing Machinery.

[227] Alannah Oleson and Amy J. Ko. Toward the Development of HCI Pedagogical Content Knowledge. page 6, 2020.

[228] Alannah Oleson, Christopher Mendez, Zoe Steine-Hanson, Claudia Hilderbrand, Christopher Perdriau, Margaret Burnett, and Amy J. Ko. Pedagogical Content Knowledge for Teaching Inclusive Design. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*, ICER '18, pages 69–77, New York, NY, USA, 2018. ACM.

[229] Alannah Oleson, Meron Solomon, and Amy J. Ko. Computing Students' Learning Difficulties in HCI Education. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–14, Honolulu, HI, USA, April 2020. Association for Computing Machinery.

[230] Alannah Oleson, Meron Solomon, Christopher Perdriau, and Amy J. Ko. Teaching Inclusive Design Skills with the CIDER Assumption Elicitation Technique. *ACM Transactions on Computer-Human Interaction*, July 2022. Just Accepted.

[231] Alannah Oleson, Brett Wortzman, and Amy J. Ko. On the Role of Design in K-12 Computing Education. *ACM Transactions on Computing Education*, 21(1):2:1–2:34, January 2021.

[232] Paul Oquist. The Epistemology of Action Research. *Acta Sociologica*, 21(2):143–163, April 1978. Publisher: SAGE Publications Ltd.

[233] Hye Park and Seda McKilligan. A Systematic Literature Review for Human-Computer Interaction and Design Thinking Process Integration. In Aaron Marcus and Wentao Wang, editors, *Design, User Experience, and Usability: Theory and Practice*, Lecture Notes in Computer Science, pages 725–740. Springer International Publishing, 2018.

[234] Arno Pasternak. Contextualized Teaching in the Lower Secondary EducationLong-term Evaluation of a CS Course from Grade 6 to 10. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 657–662, New York, NY, USA, 2016. ACM.

[235] Michael Quinn Patton. *Qualitative Research & Evaluation Methods: Integrating Theory and Practice*. SAGE Publications, October 2014.

[236] Anne-Kathrin Peters. Students' Experience of Participation in a Discipline - A Longitudinal Study of Computer Science and IT Engineering Students. *ACM Transactions on Computing Education (TOCE)*, 19(1):5:1–5:28, September 2018.

[237] Marian Petre and Andre Van Der Hoek. *Software Designers in Action: A Human-Centric Look at Design Work*. Chapman & Hall/CRC, 1st edition, 2013.

[238] Marian Petre and André van der Hoek. Beyond Coding: Toward Software Development Expertise. *XRDS*, 25(1):22–26, October 2018.

[239] Henry Petroski. *Success Through Failure: The Paradox of Design*. Princeton University Press, 2006.

[240] Feniosky Peña-Mora and Sanjeev Vadhavkar. Augmenting design patterns with design rationale. *AI EDAM*, 11(2):93–108, April 1997.

[241] Peter Pirolli and Stuart Card. Information Foraging in Information Access Environments. In *Chi*, volume 95, pages 51–58, 1995.

[242] Michael J. Prince and Richard M. Felder. Inductive Teaching and Learning Methods: Definitions, Comparisons, and Research Bases. *Journal of Engineering Education*, 95(2):123–138, 2006.

[243] Chris Proctor, Maxwell Bigman, and Paulo Blikstein. Defining and Designing Computer Science Education in a K12 Public School District. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 314–320, New York, NY, USA, 2019. ACM. event-place: Minneapolis, MN, USA.

[244] Cynthia Putnam, Maria Dahman, Emma Rose, Jinghui Cheng, and Glenn Bradford. Teaching Accessibility, Learning Empathy. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility - ASSETS '15*, pages 333–334, Lisbon, Portugal, 2015. ACM Press.

[245] Yolanda A. Rankin and Jakita O. Thomas. Straighten Up and Fly Right: Rethinking intersectionality in HCI research. *Interactions*, 26(6):64–68, October 2019.

[246] Yim Register and Amy J. Ko. Learning Machine Learning with Personal Data Helps Stakeholders Ground Advocacy Arguments in Model Mechanics. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ICER '20, pages 67–78, New York, NY, USA, August 2020. Association for Computing Machinery.

[247] Yolanda Jacobs Reimer and Sarah A. Douglas. Teaching HCI design with the studio approach. *Computer science education*, 13(3):191–205, 2003.

[248] Lauren Rich, Heather Perry, and Mark Guzdial. A CS1 Course Designed to Address Interests of Women. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '04, pages 190–194, New York, NY, USA, 2004. ACM.

[249] Horst W. J. Rittel. The Reasoning of Designers. page 12, 1987.

[250] Horst W. J. Rittel and Melvin M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155–169, June 1973.

[251] Karen Rosenblum and Toni-Michelle Travis. The Meaning of Difference: American Constructions of Race and Ethnicity, Sex and Gender. *Social Class, Sexuality, and Disability*, 2015.

[252] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O. Wobbrock. An Epidemiology-inspired Large-scale Analysis of Android App Accessibility. *ACM Transactions on Accessible Computing*, 13(1):4:1–4:36, April 2020.

[253] Margault Sacre and Carine Lallemand. Applying the TPACK model to HCI Education: Relationships between Perceived Instructional Quality and Teacher Knowledge. In *Proceedings of the 5th Annual Symposium on HCI Education*, EduCHI '23, pages 33–42, New York, NY, USA, April 2023. Association for Computing Machinery.

[254] Luciana Salgado, Roberto Pereira, and Isabela Gasparini. Cultural Issues in HCI: Challenges and Opportunities. In Masaaki Kurosu, editor, *Human-Computer Interaction: Design and Evaluation*, Lecture Notes in Computer Science, pages 60–70. Springer International Publishing, 2015.

[255] Morgan Klaus Scheuerman, Kandrea Wade, Caitlin Lustig, and Jed R. Brubaker. How We've Taught Algorithms to See Identity: Constructing Race and Gender in Image Databases for Facial Analysis. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW1):058:1–058:35, May 2020.

[256] Ari Schlesinger, W. Keith Edwards, and Rebecca E. Grinter. Intersectional HCI: Engaging Identity through Gender, Race, and Class. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 5412–5427, New York, NY, USA, May 2017. Association for Computing Machinery.

[257] Donald A. Schön. *The Reflective Practitioner: How Professionals Think In Action*. Basic Books, September 1984.

[258] Donald A. Schön. Educating the reflective practitioner. 1987.

[259] Niral Shah and Aman Yadav. Racial Justice Amidst the Dangers of Computing Creep: A Dialogue. *TechTrends*, 67(3):467–474, May 2023.

[260] Tom Shakespeare. The social model of disability. *The disability studies reader*, 2:197–204, 2006.

[261] R. Benjamin Shapiro, Kayla DesPortes, and Betsy DiSalvo. Improving Computing Education Research through Valuing Design. *Communications of the ACM*, 66(8):24–26, July 2023.

[262] Mary Shaw and David Garlan. *Software architecture*, volume 101. Prentice Hall Englewood Cliffs, 1996.

[263] Weishi Shi, Saad Khan, Yasmine El-Glaly, Samuel Malachowsky, Qi Yu, and Daniel E. Krutz. Experiential learning in computing accessibility education. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, ICSE '20, pages 250–251, New York, NY, USA, October 2020. Association for Computing Machinery.

[264] Weishi Shi, Heather Moses, Qi Yu, Samuel Malachowsky, and Daniel E. Krutz. ALL: Supporting Experiential Accessibility Education and Inclusive Software Development. *ACM Transactions on Software Engineering and Methodology*, September 2023. Just Accepted.

[265] Kristen Shinohara, Saba Kawas, Amy J. Ko, and Richard E. Ladner. Who Teaches Accessibility? A Survey of U.S. Computing Faculty. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, pages 197–202, New York, NY, USA, February 2018. Association for Computing Machinery.

[266] Lee Shulman. Knowledge and teaching: Foundations of the new reform. *Harvard educational review*, 57(1):1–23, 1987.

[267] David Siegel and Susan Dray. The map is not the territory: empathy in design. *Interactions*, 26(2):82–85, February 2019.

[268] Martin A Siegel and Erik Stolterman. Metamorphosis: Transforming Non-Designers into Designers. volume 378, pages 1–13, Sheffield, UK: Sheffield Hallam University, 2008.

[269] Siang Kok Sim and Alex H. B. Duffy. Towards an ontology of generic engineering design activities. *Research in Engineering Design*, 14(4):200–223, November 2003.

[270] Herbert A. Simon. The structure of ill structured problems. *Artificial Intelligence*, 4(3):181–201, December 1973.

[271] Jaisie Sin, Cosmin Munteanu, Michael Nixon, Velian Pandeliev, Garreth W. Tigwell, Kristen Shinohara, Anthony Tang, and Steve Szigeti. Uncovering inclusivity gaps in design pedagogy through the digital design marginalization framework. *Frontiers in Computer Science*, 4:822090, July 2022.

[272] Susan Singer and Karl A. Smith. Discipline-Based Education Research: Understanding and Improving Learning in Undergraduate Science and Engineering. *Journal of Engineering Education*, 102(4):468–471, October 2013.

[273] Karl A. Smith, Sheri D. Sheppard, David W. Johnson, and Roger T. Johnson. Pedagogies of Engagement: Classroom-Based Practices. *Journal of Engineering Education*, 94(1):87–101, January 2005.

[274] Penny Sparke and Fiona Fisher. *The Routledge Companion to Design Studies*. Routledge, June 2016.

[275] Elizabeth Starkey, Christine A. Toh, and Scarlett R. Miller. Abandoning creativity: The evolution of creative ideas in engineering design course projects. *Design Studies*, 47:47–72, November 2016.

[276] Ernest T. Stringer. *Action Research (3rd ed.)*. Sage Publications: Thousand Oaks, Californie, 2007.

[277] Simone Stumpf, Anicia Peters, Shaowen Bardzell, Margaret Burnett, Daniela Busse, Jessica Cauchard, and Elizabeth Churchill. Gender-Inclusive HCI Research and Design: A Conceptual Review. *Foundations and Trends® in Human–Computer Interaction*, 13(1):1–69, March 2020. Publisher: Now Publishers, Inc.

[278] Vanessa Svihla, Yan Chen, and Sung “Pil” Kang. A funds of knowledge approach to developing engineering students’ design problem framing skills. *Journal of Engineering Education*, 111(2):308–337, 2022. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jee.20445.

[279] Hironobu Takagi, Shinya Kawanaka, Masatomo Kobayashi, Takashi Itoh, and Chieko Asakawa. Social accessibility: achieving accessibility through collaborative metadata authoring. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, Assets ’08, pages 193–200, New York, NY, USA, October 2008. Association for Computing Machinery.

[280] Matti Tedre, Simon, and Lauri Malmi. Changing aims of computing education: a historical survey. *Computer Science Education*, 0(0):1–29, June 2018.

[281] The Association for Computing Machinery (ACM). Computing Curricula 2005: The Overview Report, 2019.

[282] Charles Thevathayan and Margaret Hamilton. Imparting Software Engineering Design Skills. In *Proceedings of the Nineteenth Australasian Computing Education Conference*, ACE '17, pages 95–102, New York, NY, USA, 2017. ACM.

[283] Harold Thimbleby. Teaching and Learning HCI. In Constantine Stephanidis, editor, *Universal Access in Human-Computer Interaction. Addressing Diversity*, Lecture Notes in Computer Science, pages 625–635. Springer Berlin Heidelberg, 2009.

[284] Debra Thompson. What Lies Beneath: Equality and the making of racial classifications. *Social Philosophy & Policy*, 31(2):114, 2015. Publisher: Cambridge University Press.

[285] Transparent Statistics in Human-Computer Interaction Working Group. Transparent Statistics Guidelines, June 2019., June 2019. https://transparentstats.github.io/guidelines.

[286] Nicholas True, Jeroen Peeters, and Daniel Fallman. Confabulation in the Time of Transdisciplinarity: Reflection on HCI Education and a Call for Conversation. In Masaaki Kurosu, editor, *Human-Computer Interaction. Human-Centred Design Approaches, Methods, Tools, and Environments*, Lecture Notes in Computer Science, pages 128–136. Springer Berlin Heidelberg, 2013.

[287] Sherry Turkle and Seymour Papert. Epistemological Pluralism: Styles and Voices within the Computer Culture. *Signs: Journal of Women in Culture and Society*, 16(1):128–157, October 1990.

[288] Sepehr Vakil. Ethics, Identity, and Political Vision: Toward a Justice-Centered Approach to Equity in Computer Science Education. *Harvard Educational Review*, 88(1):26–52, March 2018.

[289] Anna Vallgarda and Ylva Fernaeus. Interaction Design As a Bricolage Practice. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '15, pages 173–180, New York, NY, USA, 2015. ACM.

[290] András Vargha and Harold D. Delaney. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000. Publisher: Sage Publications Sage CA: Los Angeles, CA.

[291] Swaroop S. Vattam, Ashok K. Goel, Spencer Rugaber, Cindy E. Hmelo-Silver, Rebecca Jordan, Steven Gray, and Suparna Sinha. Understanding Complex Natural Systems by Articulating Structure-Behavior-Function Models. *Journal of Educational Technology & Society*, 14(1):66–81, 2011.

[292] Rebecca Vivian and Katrina Falkner. Identifying Teachers' Technological Pedagogical Content Knowledge for Computer Science in the Primary Years. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, pages 147–155, New York, NY, USA, 2019. ACM. event-place: Toronto ON, Canada.

[293] Annalu Waller, Vicki L. Hanson, and David Sloan. Including accessibility within and beyond undergraduate computing courses. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, Assets '09, pages 155–162, New York, NY, USA, October 2009. Association for Computing Machinery.

[294] Xiaowei Wang, John Mylopoulos, Giancarlo Guizzardi, and Nicola Guarino. How software changes the world: The role of assumptions. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–12, June 2016. ISSN: 2151-1357.

[295] Christopher Rhys Watkins, Colin M. Gray, Austin L. Toombs, and Paul Parsons. Tensions in Enacting a Design Philosophy in UX Practice. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, DIS '20, pages 2107–2118, New York, NY, USA, July 2020. Association for Computing Machinery.

[296] Max Weber. *From Max Weber: essays in sociology*. Routledge, 2013.

[297] Jane Webster and Richard T Watson. Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, pages xiii–xxiii, 2002.

[298] Cathleen Wharton, John Rieman, Clayton Lewis, and Peter Polson. The Cognitive Walkthrough Method: A Practitioner's Guide. In *Usability Inspection Methods*, pages 105–140. John Wiley & Sons, Inc., USA, June 1994.

[299] Cathleen Wharton, John Rieman, Clayton Lewis, and Peter Polson. Usability Inspection Methods. pages 105–140. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[300] Eliane S. Wiese, Jason Wiese, Marina Kogan, and Joshua Dawson. Lightweight Methods for Developing Pedagogical Content Knowledge for HCI. page 7, New Orleans, LA, USA, 2022.

[301] Wikipedia. Dog whistle (politics), September 2023. https://en.wikipedia.org/w/index.php?title=Dog_whistle_(politics)&oldid=1175273360.

[302] Lauren Wilcox, Betsy DiSalvo, Dick Henneman, and Qiaosi Wang. Design in the HCI Classroom: Setting a Research Agenda. In *Proceedings of the 2019 on Designing Interactive Systems Conference*, DIS '19, pages 871–883, New York, NY, USA, 2019. ACM.

[303] Michele A. Williams, Erin Buehler, Amy Hurst, and Shaun K. Kane. What Not to Wearable: using participatory workshops to explore wearable device form factors for blind users. In *Proceedings of the 12th Web for All Conference on - W4A '15*, pages 1–4, Florence, Italy, 2015. ACM Press.

[304] Amy Wilson-Lopez, Joel Alejandro Mejia, Indhira María Hasbún, and G. Sue Kasun. Latina/o Adolescents' Funds of Knowledge Related to Engineering. *Journal of Engineering Education*, 105(2):278–311, 2016. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jee.20117.

[305] Amy Wilson-Lopez, Christina Sias, Allen Smithee, and Indhira María Hasbún. Forms of science capital mobilized in adolescents' engineering projects. *Journal of Research in Science Teaching*, 55(2):246–270, 2018. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/tea.21418.

[306] D. E. Wittkower. Disaffordances And Dysaffordances In Code. *AoIR Selected Papers of Internet Research*, October 2017.

[307] Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. Ability-Based Design: Concept, Principles and Examples. *ACM Transactions on Accessible Computing*, 3(3):9:1–9:27, April 2011.

[308] Tracee Vetting Wolf, Jennifer A. Rode, Jeremy Sussman, and Wendy A. Kellogg. Dispelling "Design" As the Black Art of CHI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 521–530, New York, NY, USA, 2006. ACM.

[309] Richmond Y. Wong and Tonya Nguyen. Timelines: A World-Building Activity for Values Advocacy. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, number 616, pages 1–15. Association for Computing Machinery, New York, NY, USA, May 2021.

[310] Peter Wright and John McCarthy. Empathy and Experience in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 637–646, New York, NY, USA, 2008. ACM.

[311] Benjamin Xie, Alannah Oleson, Jayne Everson, and Amy J. Ko. Surfacing Equity Issues in Large Computing Courses with Peer-Ranked, Demographically-Labeled Student Feedback. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW1):65:1–65:39, April 2022.

[312] Aman Yadav and Marc Berges. Computer Science Pedagogical Content Knowledge: Characterizing Teacher Performance. *ACM Trans. Comput. Educ.*, 19(3):29:1–29:24, May 2019.

[313] Aman Yadav, Marie Heath, and Anne Drew Hu. Toward justice in computer science through community, criticality, and citizenship. *Communications of the ACM*, 65(5):42–44, April 2022.

[314] Sarita Yardi and Amy Bruckman. What is Computing?: Bridging the Gap Between Teenagers' Perceptions and Graduate Students' Experiences. In *Proceedings of the Third International Workshop on Computing Education Research*, ICER '07, pages 39–50, New York, NY, USA, 2007. ACM.

[315] Jason C. Yip, Kiley Sobel, Caroline Pitt, Kung Jin Lee, Sijin Chen, Kari Nasu, and Laura R. Pina. Examining Adult-Child Interactions in Intergenerational Participatory Design. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 5742–5754, New York, NY, USA, 2017. ACM.

[316] Helen Z. Zhang, Charles Xie, and Saeid Nourian. Are their designs iterative or fixated? Investigating design patterns from student digital footprints in computer-aided design software. *International Journal of Technology and Design Education*, 28(3):819–841, September 2018.

[317] Xiaoyi Zhang, Anne Spencer Ross, and James Fogarty. Robust Annotation of Mobile Application Interfaces in Methods for Accessibility Repair and Enhancement. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 609–621, New York, NY, USA, October 2018. Association for Computing Machinery.

[318] Emilene Zitkus, Patrick Langdon, and P. John Clarkson. Inclusive design advisor: understanding the design practice before developing inclusivity tools. *Journal of Usability Studies*, 8(4):127–143, August 2013.

[319] Ortrun Zuber-Skerritt. *Action research in higher education: examples and reflections.* ERIC, 1992.