Beyond HCI: The Need for Accessibility Across the CS Curriculum

Yasmine N. Elglaly
Western Washington University
elglaly@wwu.edu

Catherine M. Baker
Creighton University
catherinebaker@creighton.edu

Anne Spencer Ross
Bucknell University
a.ross@bucknell

Kristen Shinohara
Rochester Institute of Technology
kristen.shinohara@rit.edu

ABSTRACT

Few instructors include accessibility in computing education curriculum despite its importance in the computing field. Prior work on accessibility and CS education vary on what accessibility knowledge is appropriate for future computing professionals and is covered mainly in Human-Computer Interaction (HCI) and web design courses. In practice, there is no comprehensive list of learning objectives to characterize Accessibility as a competency even though it is an increasingly desired skillset across computing broadly. Moreover, in the soon-to-be-released CS2023, the Accessibility Knowledge Unit is nested within HCI. This placement fails to highlight the role accessibility plays throughout the CS curriculum, nor does it provide sufficient structure to guide CS educators. Thus, these efforts may not be enough to prepare future tech professionals with adequate accessibility skills to meet growing industry demand. We argue that Accessibility should be its own Knowledge Area within the CS curriculum. We present a set of knowledge units with topics and illustrative learning objectives for an Accessibility Knowledge Area in Computing Education. These objectives were produced through synthesizing computing education and accessible computing literature and discussions among the authors, who are subject matter experts in both computing education and accessibility. In this position paper, we make the case that Accessibility should be incorporated into computer science across the curriculum beyond just HCI, and we demonstrate how the knowledge units spans the different facets of computing. In addition to highlighting the cross-curriculum importance of Accessibility, the presented Knowledge Area outline provides structure for future work in creating teaching resources and guiding curricular integration.

CCS CONCEPTS

• **Human-centered computing → Accessibility theory, concepts and paradigms**.

KEYWORDS

Accessibility, curriculum, CS education

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

ACM ISBN 979-8-4007-0423-9/24/03…$15.00
https://doi.org/10.1145/3626252.3630788

ACM Reference Format:
Yasmine N. Elglaly, Catherine M. Baker, Anne Spencer Ross, and Kristen Shinohara. 2024. Beyond HCI: The Need for Accessibility Across the CS Curriculum. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626252.3630788

1 INTRODUCTION

Digital accessibility practices lead to creating technologies and digital content that is accessible to people with disabilities. These practices include the development of assistive technologies, i.e., technical solutions that are designed specifically for users with disabilities, such as screen readers. Digital accessibility also involves implementing software so that it is accessible to users with disabilities and compatible with assistive technologies. Achieving software accessibility requires a breadth of accessibility knowledge throughout the software creation lifecycle. It is crucial that all computing students learn about digital accessibility in their undergraduate computing programs across the computing curriculum.

Most computing programs place accessibility within Human-Computer Interaction (HCI) and design courses, if they address it at all [24, 51]. There is a wealth of knowledge on what and how to incorporate accessibility in digital design courses [45, 47, 50]. However, there has been limited effort in teaching accessibility to non-design students or in computing courses that are not centered around HCI [30]. This lack of comprehensive coverage of accessibility in computing programs produces computer science graduates with limited or no accessibility knowledge and skill and who do not meet growing industry demand for such competency [14, 46].

Current efforts are underway to revise the ten-year-old computer science curricular guidelines by ACM and IEEE to match innovative and cutting edge technologies and problems of the day [16]. As part of this effort, for the first time, Accessibility will be encapsulated within the HCI Knowledge Area. This inclusion marks a key step forward in recognizing the importance of Accessibility in usability and good user interface design and and reflects the increased industry need for attention to making technologies accessible. In this new curricular guidance, Accessibility comprises a Knowledge Unit outlining a fundamental learning objective of the HCI Knowledge Area. While this curricular change is important to ensuring that Accessibility is taught within computing at all, concepts covered within HCI will be limited to understanding users and issues reflecting user interface and interaction design. Meanwhile, underlying design and development require attention to Accessibility to be able to fully support Accessibility across the full functionality of the system. For example, databases storing images and videos must be

developed with fields for alternative text or captions; choosing security and verification techniques have accessibility implications such a input and timing; and operating system APIs must be compatible with assistive technology input. Moreover, building general awareness of accessibility supports building organizational maturity in accessibility. To comprehensively ensure software engineers and developers are knowledgeable about accessibility—so that they can provide adequate support for systems that underlie usability and core functionality—Accessibility must be included as a Knowledge Area in its own right, and not just a Knowledge Unit within HCI.

We pose Accessibility should be a distinct Knowledge Area within the CS curriculum. Toward that goal, we present core Accessibility Learning Objectives structured within core CS objectives, knowledge-area specific objectives, and non-core objectives. Following the model of CS2023, core CS objectives should be covered in every computer science program, knowledge-area (KA) objectives should be covered by programs that wish to specialize in the area, and non-core objectives structure guidance for providing additional depth or breadth to students. As subject matter experts in computing education and accessibility research with more than 40 years of collective experience, we formulated these learning objectives by leveraging prior work, community input, and our breadth of knowledge and experience. First, we reviewed the current state of accessibility education and desired outcomes as highlighted by the technology industry, for instance by analyzing related research to document curricular trajectories, e.g., [10, 42, 46]. Next, we defined the learning objectives in consultation with the Bloom's Taxonomy [25], which categorizes cognitive learning levels, from lower-order thinking skills (remembering, understanding) to higher-order skills (applying, analyzing, evaluating, creating). Last, we arranged the learning objectives from foundational concepts to more advanced skills or knowledge. We used the ACM 2023 curriculum terminology to describe the level of the desired learning outcomes, following the sequence of CS Core, Knowledge Area, and Non-Core.

We iteratively reviewed and refined the learning objectives. We also compiled a list of resources that can be used to achieve these learning objectives[1].

2 ACCESSIBILITY EDUCATION IN THE CS CURRICULUM

2.1 The Future of Tech is Accessible

In the US, the recently proposed ADA Title II Web and Mobile App Accessibility Rule would require state and local government entities make their web pages accessible [17], creating demand for accessibility knowledge and skills. Research increasingly shows that the tech industry desires accessible products, at least to meet legal and compliance requirements [46], but also because accessible design is considered good design. Top tech companies have recently put in concerted effort to make tech products usable by people with disabilities (e.g., Google [5], Microsoft [43], Apple [3]). While it is helpful to design user interfaces that are accessible to people with disabilities, underlying system design and development practices are often unable to meet accessible design needs. In addition, research on the landscape of tech jobs shows that few employers appear to recruit based on accessibility skills, yet they require employees to fulfill accessibility compliance tasks, and to train and mentor coworkers in accessibility knowledge and skills [42, 46]. This approach results in tech employees not primarily tasked with accessibility-focused work, yet responsible for ensuring compliance [46], manifesting as a requirements and skills gap. Ultimately, Accessibility is overlooked as a key component in tech design and development, and as a result is not allocated resources and effort.

Further complicating the issue for employers, the demand for user interface and user experience designers with accessibility skills continues to grow, but few applicants have skills and competency in Accessibility to meet that need [14]. Recent industry research found a disparity between needs and employee competencies, specifically that "there is a significant disconnect between understanding accessibility standards and creation of content that actually adheres to them" [14]. Thus, accessibility is emerging as a desired skill among tech companies who report issues recruiting job candidates with the necessary skills and know-how. This issue is compounded by the lack of exposure to accessibility for computer science graduates. For instance, few faculty report including accessibility in their computing courses, and most of those faculty are HCI experts [51]. As a result, few computing graduates are equipped with accessibility knowledge and skills [46], and even fewer have an understanding of accessibility beyond the surface.

The emphasis on user interface design and interaction is important to ensuring access for disabled users, but the underlying technology must also be built to support accessible front-end design. Accessibility can be affected by the early stages of selecting the libraries and frameworks to use for a project. Similarly, testing must be updated to incorporate accessibility to ensure the final product *is* accessible. Testing for accessibility requires knowledge about accessibility and accessible user interfaces, and it also requires knowing how to work with people with disabilities appropriately as usability testers. As we discuss below, these needs demonstrate that Accessibility skillsets are broadly applicable across computing and should not be relegated only to surface-level design choices.

2.2 Curricular Guidelines do not Comprehensively Cover Accessibility

Current computer science curriculum guidance, in place since 2013, do not explicitly include accessibility as a key knowledge area. Including Accessibility as a knowledge unit within HCI in the updated CS2023 curricular guidelines asserts accessibility as a core part of user interface and user interaction design and development and will inform curricular elements [16]. For example, these guidelines will ensure future graduates understand how people with a variety of disabilities use technology, and know about various assistive technologies that provide access to computing technology. Although inclusion in updated curricular guidelines is an important first step, restricting Accessibility to HCI disregards the need for understanding assistive technology and accessible solutions at the development and engineering phases. We emphasize that, to support accessible user interface design, underlying systems should be engineered to be accessible through low-level infrastructure to propagate accessibility via native controls and operations [48]. In

[1] https://accessibilityeducation.github.io/resources.html

the worst case, students of computing outside of HCI will not be aware of Accessibility as an important component of computing.

Despite the need to include Accessibility as a Knowledge Area across computing curriculum, there is not yet clear structure or guidance for what should be taught, what pedagogies are appropriate, and how to assess learning for Accessibility skill and competency [38]. Our Accessibility Knowledge Area outlines important topics organized by Knowledge Units. This structure helps organize existing curricular development efforts, highlights areas with particular need of material development, and guides programs in choosing topics to encorporate. Guidance from our Knowledge Area can help build computing pedagogy to meet specific tech demands for accessibility skills [38].

3 ACCESSIBILITY KNOWLEDGE AREA

We began by delineating five fundamental accessibility knowledge units: Disability Awareness, Accessibility Design, Accessibility Implementation, Accessibility Evaluation, and Accessibility Profession and Continuous Learning. These knowledge units represent clusters of competencies sought after in the technology industry [42], but that are not comprehensively covered in computing programs [24, 46]. For each knowledge unit, we outline a list of topics and for each topic we suggest illustrative learning objectives. These topics draw from related work, our experience, and ongoing discussions among researchers and industry in accessibility education workshops and panels [23, 36].

We structured the topics and their associated learning objectives in line with the format of the CS2023 Curriculum. The topics were organized into three tiers of knowledge: the CS core, the Knowledge Area (KA) core, and the Non-Core. The CS core level corresponds to the fundamental concepts essential for every computer science program. The KA core level presents the option to dive further into a specific knowledge area, enhancing the curriculum's depth and scope. Lastly, the Non-Core level outlines advanced learning for those who want to specialize in accessibility, going beyond foundational knowledge.

3.1 Knowledge Unit 1 - Disability Awareness

An essential component to digital accessibility is knowledge about disabilities, assistive technologies, and general context on disabled experiences. This unit outlines goals for exposing students to the range of disabled experiences that inform digital accessibility.

Throughout this unit, it is important to forefront disability-aware perspectives in framing disability as a wide and nuanced experience, highlighting disabled people as active contributors to their environment and communities. Disabled people are advocates, creators, and engineers, and are not just passive recipients of "charity". A robust education in digital accessibility must include understanding the experiences of disabled people. These topics are essential to motivating the need for digital accessibility, understanding accessibility best practices from a human-centered perspective, and contextualizing practices within societal systems.

(1) CS Core
   (a) **Basic categories of disabilities.** List broad categories of disability, including: vision, auditory, motor, and cognitive [20]. Understand that disability can be permanent, temporary, or situational, and that disability is a varied and nuanced experience that extends beyond these basic classifications [32].
   (b) **Motivations for accessibility.** Draw from human, legal, and business cases. Recognize accessibility benefits over one billion disabled persons [4], in addition to people with temporary and situational disabilities [7]. Recognize that current software development does not keep pace with accessibility needs [20, 48].
   (c) **Assistive technology.** Demonstrate awareness of some assistive technology like screen readers, switch devices, captions, voice command.
   (d) **Ethics**. Apply the principles outlined in the ACM Code of Ethics and Professional Conduct to assess the ethical implications of technology for people with disabilities [2].
(2) KA Core:
   (a) **Basic models of disability.** Understand that there are different models of disability, and be able to define key ones, e.g., the medical model (which views disability as problem in the person) vs. the social model (which views disability as a contextual problem that is disabling).
   (b) **Experiences of disability.** Build on the broad categories of disability to be aware of more variation and depth in user experiences. For example, knowledge of discussions on language used in disabled communities; dynamics of identifying as disabled versus not.
   (c) **Ableism and etiquette.** Be familiar with the "Nothing About Us Without Us" movement [26], specifically, that disabled people face systemic biases. Understand etiquette for interacting with people with disabilities (e.g., do not move wheelchairs, talk to people not their interpreters).
   (d) **Intersectionality.** Understand how disability intersects with other identities such as race, gender, location, socioeconomic status and how it may compound systemic injustices or change community dynamics [49].
   (e) **Use common assistive technology.** Perform basic functions using some assistive technology such as screenreaders, switch access, and digital magnifier.
   (f) **Make a case for accessibility**. Make a pitch for why accessibility is important. Draw from human, legal, business, and software inaccessibility prevalence information.
(3) Non-Core:
   (a) **Advanced models of disability.** Understand more nuanced models of disability beyond the medical and social models, such as the legal/ethical model [29]. Understand the strengths and critiques of each.
   (b) **Disability identities.** Consider broader ranges and environments of disabled experiences, for example, people who are chronically ill; discussions on language used in disabled communities; dynamics of identifying as disabled versus not [39].
   (c) **History of accessibility.** Be familiar with disabled people's roles in the evolution of the telephone, text messaging, and other technical systems. Understand how disability-led advocacy and activism contributed to passing accessibility legislation: the sit-in demonstrations to push the passing of Section 504 of the Rehabilitation Act [26].

3.2 Knowledge Unit 2 - Accessibility Design

Several approaches to technology design have been leveraged to support improved accessible design, including Universal Design [21] and Ability Based Design [55]. In practice, most accessible design work is driven by compliance and usability [46]. The recently published ADA Title II Web and Mobile App Accessibility Rule details a technical standard for web accessibility [17] and will drive a need for tech professionals with accessibility design competency to meet requirements.

Central to the Title II Web and Mobile App Accessibility Rule is the Web Content Accessibility Guidelines (WCAG), created by the World Wide Web Consortium (W3C), and which comprises a set of design recommendations to make web content accessible to people with disabilities [33]. Future tech professionals will need to be familiar with WCAG, how to meet compliance, and to understand what tools and processes enable them to design and maintain systems and apps that are compliant.

(1) CS Core
   (a) **Design guidelines and knowledge.** Describe users with disabilities using proper language. Apply W3C accessibility guidelines to design. Gain knowledge of Web Content Accessibility Guidelines, what it means to be compliant, and how to meet compliance. Become familiar with specific details such as how to choose appropriate colors, layout design, and font selection, and know how to use design tool annotations.
   (b) **Methods and approaches.** Be familiar with design methods and approaches that include accessibility. Understand how to collect software requirements from users with disabilities, e.g., ask deaf participants if they prefer a captionist or an interpreter prior to conducting usability studies.
   (c) **Assistive technology awareness.** Understand the process of designing technology specifically for people with disabilities as the main target users.
(2) KA Core:
   (a) **Design accessible software.** Apply web and mobile accessibility guidelines to make new designs accessible, and update legacy designs. Design software that can be customized. Distinguish between different design approaches, including the Principles of Universal Design and how they may be used for different kinds of disabilities. Apply industry accessibility guidelines, (e.g., from Google and Microsoft).
   (b) **Improve assistive technology**. Apply a design process to improve an existing assistive technology. Distinguish between assistive technology-first designs and those that are not.
(3) Non-Core:
   (a) **Special topics.** Understand XR (AR/VR) accessibility guidelines, know how accessible design can inform AI-powered systems. Understand design implications for users of systems powered by artificial intelligence.
   (b) **Design assistive technology.** Apply a design process to create a new assistive technology with people with disabilities.

3.3 Knowledge Unit 3 - Accessibility Implementation

This unit has two primary facets: proficiency in creating software accessible to disabled people and implementing assistive technology. Accessibility implementation knowledge is key to enabling developers to build accessible software, i.e., software that can be used by people with disabilities either directly or through the means of assistive technology. Most assistive technologies include software components and many of them are completely digital. Given that developers are pivotal in crafting assistive technology, it is imperative for computing students to learn about the relationship between software design, accessible software development, and how systems can support assistive technologies.

(1) CS Core
   (a) **Accessibility requirements.** Identify and comprehend specific accessibility requirements relevant to users with disabilities.
   (b) **Software architecture.** Understand the impact of software architecture on accessibility. Evaluate and select frameworks and libraries that support accessibility features and conform to accessibility standards [40].
   (c) **Software implementation.** Select templates or starter code that follow accessibility best practices. For example, the default mobile application in Android Studio includes accessible coding practices. Customize accessible templates for specific requirements while maintaining the accessibility.
   (d) **Assistive technology architecture.** Understand the overall architecture of one example of assistive technology. Understand how an assistive technology, e.g., a screen reader, interfaces with operating systems and web browsers.
(2) KA Core
   (a) **Accessibility guidelines.** Where appropriate, apply accessibility guidelines (such as WCAG) to developed software. For example, implement keyboard navigation so that contents are accessed in a logical order.
   (b) **ARIA: Accessible Rich Internet Applications.** Understand the purpose of ARIA and explain the role of ARIA attributes in enhancing the accessibility of web content. Identify common ARIA roles such as roles for buttons, links, landmarks, and attributes like aria-label, aria-labelledby, and aria-describedby. Understand when ARIA should and should not be applied.
   (c) **Assistive technology code comprehension.** Analyze the code components of an open source assistive technology, such as NVDA, the open source screen reader [8].
   (d) **Assistive technology configuration files.** Locate the various assistive technology setting and configuration files for an assistive technology within an operating system. Understand the meaning of the different attributes and values of assistive technology settings.
(3) Non-Core
   (a) **Implement accessibility tools.** Develop tools that aid in the creation of accessible software. For example, create an accessibility linter that can analyze code for accessibility

bugs and generate a report to developers. Ensure these tools are also accessible.
(b) **Accessibility APIs.** Be knowledgeable of and know how to use systems' accessibility API, such as Apple accessibility API [3] and Java accessibility API [11].
(c) **Accessibility documentation.** Outline the architectural decisions and strategies taken to achieve accessibility goals. Document accessibility features in a software project and communicate these features to stakeholders.
(d) **Assistive technology implementation.** Implement an improvement to an existing assistive technology. For example, implement an extension or add-on to NVDA.
(e) **Assistive technology hardware interfaces.** Understand assistive technology hardware, e.g., adaptive keyboard [6], and how they connect to and interact with standard protocols, e.g., Bluetooth and USB. Specifically, understand how the operating system and software applications recognize and work with adaptive tools.
(f) **Assistive technology documentation.** Create documentation detailing the components and functionalities of one example of an assistive technology.

3.4 Knowledge Unit 4 - Accessibility Evaluation

Software products must be tested for accessibility to ensure it can be used effectively by individuals with disabilities, either directly or using an assistive technology. Accessibility evaluation involves testing software using a range of means, including automated, manual, and testing with people with disabilities. It also includes testing assistive technology for functional and non-functional requirements. Beyond the US, there is a growing global acknowledgement of accessibility laws and regulations that mandate accessibility for digital products [17, 18]. Adhering to these regulations necessitates proficiency in effective testing methods. CS curriculum should encompass knowledge and skills in accessibility testing that not only meets legal requirements, but also actively seeks disabled users' feedback and that enhances usability for all users.

(1) CS Core
(a) **Automated accessibility testing.** Recognize and use automated accessibility testing tools, such as the open source accessibility testing engine Axe [15] and the web evaluation suite WAVE [12].
(b) **Testing with an assistive technology.** Evaluate software accessibility by using an assistive technology, e.g., screen reader and a keyboard for navigation.
(c) **Usability testing.** Define usability testing and explain its significance in evaluating software accessibility from the perspective of individuals with disabilities. Specify techniques for collecting feedback from users with disabilities during testing, including surveys and interviews.
(d) **Write accessibility reports.** Document testing results to be understandable and actionable.
(2) KA Core
(a) **Testing tools.** Compare automated testing tools and understand their limitations. Set up and configure accessibility testing tools to assess websites, applications, and documents.
(b) **Testing reports.** Interpret automated accessibility testing reports. Understand the potential impact of issues on users. Document needed improvements based on the test results.
(c) **Unit testing.** Write test cases that catch accessibility defects, e.g., write a unit test to detect UI elements missing alt text.
(d) **User tasks.** Design user tasks that represent the typical interactions users perform using assistive technology such as switch devices or voice recognition.
(e) **Usability of assistive technology.** Evaluate the usability of a few assistive technologies, whether they are software or hardware devices.
(3) Non-Core
(a) **Test planning.** Create a test plan that compliments automated testing tools with other testing approaches, and that evaluates the software for users with different types of disabilities, e.g., testing plan for blind users, or for users with motor disabilities.
(b) **Accessibility testing integration.** Integrate automated accessibility tests into continuous integration and continuous deployment (CI/CD) pipelines.
(c) **Test results.** Prioritize and remediate detected accessibility issues.
(d) **Assistive technology testing.** Test assistive technologies for bugs. Evaluate the usability of an existing assistive technology with users with disabilities. Evaluate non-functional requirements of assistive technologies such as security and performance.

3.5 Knowledge Unit 5 - Accessibility Profession and Continuous Learning

Within software professions, accessibility skills are required for both general software professions (e.g., developer, tester), as well as accessibility-specialized roles [42]. Consequently, it is imperative for CS curricula to overview the breadth and depth of the required skills in these two contexts. Furthermore, maintaining up-to-date accessibility knowledge is important as accessibility standards, tools, and assistive technologies continues to evolve. The emergence of new technologies bring opportunities and challenges for people with disabilities requiring constant consideration by software professionals. Therefore, continuous and extracurricular accessibility learning shall be integrated into the CS curriculum.

(1) CS Core
(a) **Professional accessibility skills.** Understand the accessibility skills expected in general technology roles. Recognize the need to work collaboratively with accessibility experts and people with disabilities to ensure accessible technology.
(b) **Accessibility of emerging technology.** Understand the accessibility support and limitations of emerging technologies, such as AI-based systems, virtual reality, and wearable devices. Apply established accessibility principles and guidelines to emerging technologies, ensuring inclusion from the outset.
(2) KA Core

(a) **Accessibility specialization.** Differentiate between accessibility-specialized positions in industry and their required skills [14, 22].
(b) **Awareness of accessibility and disability communities.** Identify and locate local and online accessibility advocacy groups, disability-focused communities, organizations, and forums.
(c) **Evaluate accessibility of emerging technology.** Identify the accessibility implications of emerging technologies. Recognize the unique accessibility challenges that emerging technologies pose for disabled people.

(3) Non-Core
(a) **Accessibility advocacy.** Develop advocacy skills to promote accessibility awareness and integration in organizations and projects.
(b) **Accessibility certification.** Recognize accessibility professional certifications [28, 54] and are required for some accessibility expert positions [42].
(c) **Community service**. Understand the role of computing in community service with focus on the disability community. Create a plan for connecting the computing professionals with disability communities.
(d) **Improve accessibility of emerging technology.** Create a plan to address accessibility limitations in emerging technologies for users with disabilities.
(e) **Continuous learning.** Be familiar with accessibility academic and professional resources and communities to stay informed about evolving accessibility standards and tools [1, 9, 19].

4 DISCUSSION

4.1 Evolution of Accessibility Learning

Accessibility education in computing started as stand alone assignments [41] and single courses [30]. It has evolved to become part of some HCI and web design courses [16, 53], and in rare cases, a part of the curriculum [52]. These efforts show the enduring need for accessibility education and the benefits of accessibility in industry and CS departments [31]. For these reasons, we expect comprehensive coverage of accessibility knowledge and skills across the CS curriculum is necessary and viable [23]. Further, this integration is poised to bridge the accessibility skills gap between industry needs and academic outcomes [46], while concurrently enhancing the overall climate of inclusion of CS programs, particularly for students from underrepresented backgrounds [31].

The proposed learning objectives serve as a foundational blueprint for integrating accessibility across the CS curriculum. This outline provides structure for the ongoing efforts to develop curricular elements and integrate them into courses. In other words, this document outlines the *what* of accessibility across CS and can structure efforts to explore *how* to integrate into the curriculum. These objectives will need to be updated to align with evolving technology and the changing requirements of individuals with disabilities. Additionally, it will be necessary to revisit how objectives are mapped to fundamental and advanced domains among ever-changing accessibility tools, which may impact the level of mastery required. Continuous engagement among computing educators, industry experts, accessibility and disability researchers, and advocates should drive ongoing refinement of these learning objectives.

4.2 Achieving the Accessibility Learning Objectives

Effectively weaving accessibility throughout CS courses across the curriculum necessitates systemic institutional endorsement. General computing programs are encouraged to, at minimum, integrate the core accessibility learning objectives into their courses. Then, they can add the knowledge area and non-core accessibility learning objectives to their selected concentrations accordingly. For some of the accessibility learning objectives described in this paper, there exist suggestions regarding their potential integration within specific CS courses [37, 47]. However, there remains a need for further research to determine the most appropriate "home courses" for various accessibility learning objectives, particularly those that might not necessarily align with HCI courses. Our Knowledge Area helps motivate and guide those endeavors.

It is equally important that CS instructors receive training to effectively teach accessibility topics. Faculty training can be through professional development opportunities [35, 44]. Additionally, several ongoing projects are working on producing accessibility teaching materials [13, 19], which CS instructors can choose from and adapt for their courses. In these efforts, HCI-related accessibility learning objectives were used alongside various pedagogical approaches such lectures, projects, and games [24, 27, 45, 47]. For non-HCI focused learning objectives, current accessibility education research suggests using programming assignments to infuse accessibility into existing core CS courses such as data structures and computer systems [34]. More research is needed to assess the most effective pedagogical approaches for achieving accessibility learning outcomes in core courses.

5 CONCLUSION

This paper underscores the necessity of comprehensively teaching accessibility in computer science curricula, beyond HCI. We offer a set of knowledge units with example learning objectives that are based on industry demands and aligned with the CS2023 curriculum model. We acknowledge the road ahead is multifaceted, requiring collaboration between academia, industry, and the disability community. Adopting these learning objectives can bridge the accessibility skills gap evident between industry needs and academic outcomes and promote more inclusive computing education.

6 ACKNOWLEDGMENT

This work is supported by the United States National Science Foundation under grants #2121606, #2121428, #2121549, Henry Luce Foundation - Clare Boothe Luce Fund, and Swanson Fellowship from Bucknell University.

REFERENCES

[1] [n. d.]. AccessComputing. https://www.washington.edu/accesscomputing/
[2] [n. d.]. *ACM Code of Ethics and Professional Conduct.* https://www.acm.org/code-of-ethics
[3] [n. d.]. Apple Accessibility. Make your apps accessible to everyone who uses Apple devices. https://developer.apple.com/documentation/accessibility

[4] [n. d.]. Disability. World Health Organization. https://www.who.int/health-topics/disability
[5] [n. d.]. Google Accessibility. https://www.google.com/accessibility/
[6] [n. d.]. Keyboard and mouse alternatives and adaptations. https://abilitynet.org.uk/factsheets/keyboard-and-mouse-alternatives-and-adaptations
[7] [n. d.]. Microsoft Inclusive Design. https://inclusive.microsoft.design/
[8] [n. d.]. NV Access. https://www.nvaccess.org/
[9] [n. d.]. Special Interest Group on Accessible Computing. https://www.sigaccess.org/
[10] [n. d.]. Teach Access Accessibility Skills Hiring Toolkit. https://teachaccess.github.io/accessibility-skills-hiring-toolkit/
[11] [n. d.]. The Java Accessibility API. https://docs.oracle.com/cd/E17802_01/j2se/javase/technologies/accessibility/docs/jaccess-1.3/doc/core-api.html
[12] [n. d.]. WAVE Web Accessibility Evaluation Tools. https://wave.webaim.org/
[13] 2021. Including Accessibility in CS Education. https://accessibilityeducation.github.io/
[14] 2022. Teach Access. Accessible Technology Skills Gap. https://teachaccess.org/accessibility-skills-gap/
[15] 2023. Axe-core. Accessibility engine for automated Web UI testing. https://github.com/dequelabs/axe-core original-date: 2015-06-10T15:26:45Z.
[16] 2023. Computer Science Curricula 2023. CS2023 Version Beta. https://csed.acm.org/cs2023-beta/
[17] 2023. Fact Sheet: Notice of Proposed Rulemaking on Accessibility of Web Information and Services of State and Local Government Entities. https://www.ada.gov/notices/2023/07/20/web-nprm/
[18] 2023. *Global Law and Policy.* Retrieved 2023-08-18 from https://www.lflegal.com/global-law-and-policy/
[19] 2023. Teach Access website. https://teachaccess.org/
[20] 2023. WebAIM: The WebAIM Million - The 2023 report on the accessibility of the top 1,000,000 home pages. https://webaim.org/projects/million/
[21] Demosthenes Akoumianakis and Constantine Stephanidis. 1989. Universal Design in HCI: A critical review of current research and practice. *Engineering and Construction* 754 (1989).
[22] Shiri Azenkot, Margot J. Hanley, and Catherine M. Baker. 2021. How Accessibility Practitioners Promote the Creation of Accessible Products in Large Companies. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (April 2021), 1–27. https://doi.org/10.1145/3449222
[23] Catherine M. Baker, Yasmine N. Elglaly, Anne Spencer Ross, and Kristen Shinohara. 2022. Including Accessibility in Computer Science Education. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '22)*. ACM, New York, NY, USA, 1–5. https://doi.org/10.1145/3517428.3550404
[24] Catherine M. Baker, Yasmine N. Elglaly, and Kristen Shinohara. 2020. A Systematic Analysis of Accessibility in Computing Education Research. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 107–113. https://doi.org/10.1145/3328778.3366843
[25] Benjamin S Bloom, Max D Engelhart, EJ Furst, Walker H Hill, and David R Krathwohl. 1956. Handbook I: cognitive domain. *New York: David McKay* (1956).
[26] James I. Charlton. 1998. *Nothing About Us Without Us: Disability Oppression and Empowerment.* University of California Press. Google-Books-ID: ohqff8DBt9gC.
[27] Paula Conn, Taylor Gotfrid, Qiwen Zhao, Rachel Celestine, Vaishnavi Mande, Kristen Shinohara, Stephanie Ludi, and Matt Huenerfauth. 2020. Understanding the Motivations of Final-year Computing Undergraduates for Considering Accessibility. *ACM Transactions on Computing Education* 20, 2 (June 2020), 1–22. https://doi.org/10.1145/3381911
[28] CPACC [n. d.]. Certified Professional in Accessibility Core Competencies. https://www.accessibilityassociation.org/s/certified-professional
[29] Lennard J. Davis. 2010. *Disability Studies Reader* (third edition ed.). Taylor and Francis, New York, NY.
[30] Yasmine N. Elglaly. 2020. Teaching Accessibility to Software Engineering Students. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. ACM, New York, NY, USA, 121–127. https://doi.org/10.1145/3328778.3366914
[31] Rosalinda Garcia, Patricia Morreale, Lara Letaw, Amreeta Chatterjee, Pankati Patel, Sarah Yang, Isaac Tijerina Escobar, Geraldine Jimena Noa, and Margaret Burnett. 2023. "Regular" CS × Inclusive Design = Smarter Students and Greater Diversity. *ACM Transactions on Computing Education* 23, 3 (Sept. 2023), 1–35. https://doi.org/10.1145/3603535
[32] Megan Hofmann, Devva Kasnitz, Jennifer Mankoff, and Cynthia L Bennett. 2020. Living Disability Theory: Reflections on Access, Research, and Design. In *Proceedings of the 22nd International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '20)*. ACM, New York, NY, USA, 1–13. https://doi.org/10.1145/3373625.3416996
[33] W3C Web Accessibility Initiative (WAI). [n. d.]. WCAG 2 Overview. https://www.w3.org/WAI/standards-guidelines/wcag/
[34] Lin Jia, Yasmine N. Elglaly, Catherine M. Baker, and Kristen Shinohara. 2021. Infusing Accessibility into Programming Courses. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (CHI EA '21)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3411763.3451625
[35] Saba Kawas, Laura Vonessen, and Amy J. Ko. 2019. Teaching Accessibility: A Design Exploration of Faculty Professional Development at Scale. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, Minneapolis MN USA, 983–989. https://doi.org/10.1145/3287324.3287399
[36] Richard Ladner, Amy Ko Ko, and Kristen Shinohara. 2022. Integrating Accessibility and Disability into the Computing Curriculum | DO-IT. https://www.washington.edu/doit/integrating-accessibility-and-disability-computing-curriculum
[37] Richard E Ladner, Stephanie Ludi, and Robert J Domanski. 2023. Teaching about Accessibility in Computer Science Education (DRAFT 1). (2023).
[38] Sarah Lewthwaite, Sarah Horton, and Andy Coverdale. 2023. Researching Pedagogy in Digital Accessibility Education. *ACM SIGACCESS Accessibility and Computing* 134 (2023). Publisher: ACM New York, NY, USA.
[39] Simi Linton. 1998. *Claiming Disability: Knowledge and Identity.* NYU Press. Google-Books-ID: IxUVCgAAQBAJ.
[40] Michael Longley and Yasmine N Elglaly. 2021. Accessibility support in web frameworks. In *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. 1–4.
[41] Stephanie Ludi. 2007. Introducing Accessibility Requirements through External Stakeholder Utilization in an Undergraduate Requirements Engineering Course. In *29th International Conference on Software Engineering (ICSE'07)*. 736–743. https://doi.org/10.1109/ICSE.2007.46 ISSN: 1558-1225.
[42] Lilu Martin, Catherine Baker, Kristen Shinohara, and Yasmine N. Elglaly. 2022. The Landscape of Accessibility Skill Set in the Software Industry Positions. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '22)*. ACM, New York, NY, USA, 1–4. https://doi.org/10.1145/3517428.3550389
[43] Microsoft. 2018. Microsoft Design. https://www.microsoft.com/design/inclusive/
[44] Christa Miller. 2023. Accessibility Within Professional Development: Two Promising Practices. *Journal of Postsecondary Education & Disability* 36, 1 (2023). ISBN: 2379-7762.
[45] Nidhi Rajendra Palan, Vicki L. Hanson, Matt Huenerfauth, and Stephanie Ludi. 2017. Teaching Inclusive Thinking in Undergraduate Computing. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, Maryland USA, 399–400. https://doi.org/10.1145/3132525.3134808
[46] Rohan Patel, Pedro Breton, Catherine M. Baker, Yasmine N. Elglaly, and Kristen Shinohara. 2020. Why Software is Not Accessible: Technology Professionals' Perspectives and Challenges. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems (CHI EA '20)*. ACM, New York, NY, USA, 1–9. https://doi.org/10.1145/3334480.3383103
[47] Cynthia Putnam, Maria Dahman, Emma Rose, Jinghui Cheng, and Glenn Bradford. 2016. Best Practices for Teaching Accessibility in University Classrooms: Cultivating Awareness, Understanding, and Appreciation for Diverse Users. *ACM Transactions on Accessible Computing* 8, 4 (May 2016), 1–26. https://doi.org/10.1145/2831424
[48] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O Wobbrock. 2020. An epidemiology-inspired large-scale analysis of android app accessibility. *ACM Transactions on Accessible Computing (TACCESS)* 13, 1 (2020), 1–36.
[49] Sami Schalk. 2022. *Black disability politics.* Duke University Press.
[50] Kristen Shinohara, Cynthia Bennett, Jacob Wobbrock, and Wanda Pratt. 2017. Teaching Accessibility in a Technology Design Course. In *Proceedings of The 12th International Conference on Computer Supported Collaborative Learning (CSCL '17)*. International Society of the Learning Sciences, Philadelphia, PA, 239–246. https://cscl17.files.wordpress.com/2017/06/finalvol1cscl2017.pdf
[51] Kristen Shinohara, Saba Kawas, Amy J. Ko, and Richard E. Ladner. 2018. Who Teaches Accessibility? A Survey of U.S. Computing Faculty. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 197–202. https://doi.org/10.1145/3159450.3159484
[52] Annalu Waller, Vicki L. Hanson, and David Sloan. 2009. Including accessibility within and beyond undergraduate computing courses. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*. ACM, Pittsburgh Pennsylvania USA, 155–162. https://doi.org/10.1145/1639642.1639670
[53] Ye Diana Wang. 2012. A holistic and pragmatic approach to teaching web accessibility in an undergraduate web design course. In *Proceedings of the 13th annual conference on Information technology education*. ACM, Calgary Alberta Canada, 55–60. https://doi.org/10.1145/2380552.2380568
[54] WAS 2022. *Web Accessibility Specialist.* Retrieved August 18, 2023 from https://www.accessibilityassociation.org/s/wascertification
[55] Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-Based Design: Concept, Principles and Examples. *ACM Transactions on Accessible Computing* 3, 3 (April 2011), 9:1–9:27. https://doi.org/10.1145/1952383.1952384