Teaching Programming Logic for People with Blindness or Visual Impairments: a Systematic Mapping Study

Daniel S. dos Santos, Nina N. Shibata and
Victor Hugo S. C. Pinto

March 31, 2025

Teaching Programming Logic for People with Blindness or Visual Impairments: a Systematic Mapping Study

Daniel S. dos Santos
Federal University of Pará
Belém, Pará, Brazil
daniel.santos@icen.ufpa.br

Nina Niwa Shibata
Federal University of Pará
Belém, Pará, Brazil
nina.shibata@gmail.com

Victor Hugo S. Costa Pinto
Federal University of Pará
Belém, Pará, Brazil
victor.santiago@ufpa.br

Abstract

Context: Teaching programming logic is a recognized challenge, especially for people with visual impairment (PwVI) and blindness. **Problem:** There is a knowledge gap regarding the most effective methods and tools for teaching programming logic to PwVI. Additionally, the evaluation techniques used to assess the effectiveness of these approaches, especially in the context of Human-Computer Interaction (HCI), are little exploited in the literature. **Solution:** This study systematically mapped the methods and tools used in teaching programming logic to PwVI and the evaluation techniques applied in HCI. The goal is to provide an updated overview and guidance for educators and developers of educational materials. **Method:** A systematic mapping of the literature was conducted, and 13 relevant studies were selected to extract teaching methods, such as tactile flowcharts and tangible programming kits. The primary evaluation techniques identified included user testing, questionnaires, and interviews. **Summary of Results:** The teaching methods identified are diverse, emphasizing sensory resources. The most common HCI evaluation techniques helped validate the usability and effectiveness of these tools. **Contributions:** This study contributes by mapping practices and techniques for teaching programming logic to PwVI. The insights gained offer practical guidelines for educators and developers, supporting more inclusive materials and expanding accessibility within Computer Science.

CCS Concepts

• **Human-centered computing → Empirical studies in accessibility**.

Keywords

People with Visual Impairments, Programming Education, Accessibility

ACM Reference Format:
Daniel S. dos Santos, Nina Niwa Shibata, and Victor Hugo S. Costa Pinto. 2025. Teaching Programming Logic for People with Blindness or Visual Impairments: a Systematic Mapping Study. In *Proceedings of Innovation and Technology in Computer Science Education (ITiCSE'25).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

According to Guzdial et al. (2019), computational thinking is essential for problem-solving through logical structures and algorithms. Loksa et al. (2016) state that programming has become an essential skill of the 21st century, and teaching algorithms is a challenge due to the complexity involved in developing logical reasoning, especially for beginners.

With the advancement of social inclusion in computer science education, new challenges arise for educators to address the increasing presence of people with visual impairments (PwVI) in classrooms and learning environments. Furthermore, most of the materials used for teaching computational thinking are still very limited, as they emphasize the use of visual elements such as images, symbols, letters, and numbers. As a result, PwVI students internalize information during the knowledge acquisition process in a reality filled with references and patterns that are predominantly visual, which certainly places them at a disadvantage [10].

When discussing the teaching of programming logic to sighted individuals, numerous challenges arise within the classroom. Koliver et al. (2004) suggest that students' difficulty in algorithm courses is related to their lack of preparation in dealing with the logical solution of problems. Wilson (2017) indicates that these students often 'memorize' content rather than develop a deep understanding of algorithmic logic, which can worsen their learning difficulties. Arruda et al. (2024) state that teaching programming to undergraduate students often faces challenges related to their difficulty in understanding abstractions, logic, and fundamental concepts. Ribeiro Filho et al. (2024) argue that teaching programming to students with visual impairments requires adaptive strategies to overcome accessibility barriers and promote inclusion. They emphasize the need for practical methodologies, such as tactile activities, raised flowcharts, and the use of screen readers, in addition to technical and human support. Moreover, when students with visual impairments are present in algorithm classes, the challenge becomes even greater due to the students' physical limitations, teachers' lack of preparation, and the lack of accessible teaching materials [35].

Several studies proposed in the academic literature attempt to mitigate the difficulties faced by visually impaired students in algorithm classes. Tools such as Emacspeak [39] and programming systems with auditory feedback have been developed to make programming environments accessible to blind users [43]. Capovilla et al. (2013) and Papazafiropulos et al. (2016) propose the use of haptic models and 3D printing to help visually impaired students understand algorithms and data structures. Junior et al. (2009) and Ferreira et al. (2016) explored the use of podcasts as a tool for teaching algorithms, suggesting that alternative technologies, such as audio, can increase motivation for students with disabilities. Finally, Branham and Kane (2015) highlighted the importance of creating assistive technologies for visually impaired children, such as tactile and auditory feedback, to support collaboration between peers with different visual abilities.

Given the challenges and efforts to identify practical approaches for teaching algorithms to visually impaired students, a systematic literature mapping was undertaken to offer a broad overview of this research field. From the collected studies, opportunities for contributing to the improvement of algorithm teaching tools for visually impaired students were identified. However, the field still lacks more scientific evidence due to the very few works that have been published.

Moreover, the main motivation for addressing evaluation techniques in HCI in this work stems from their recognition as one of the three fundamental principles of user-centered design and a central activity in almost all design models [30]. Evaluation goes beyond verifying system functionality, contributing to creating more satisfying and efficient user experiences. Addressing system evaluations based on HCI techniques is, therefore, an opportunity to deepen the understanding of how these practices ensure usability and shape the directions of human-centered design. This work aims to explore how evaluations can be structured to promote systems that are more accessible, intuitive, and aligned with the real needs of people with visual impairments.

This study is structured as follows: section 2 presents the related works. section 3 presents the methodology used to develop the systematic literature mapping. section 4 presents the results found, with the computational thinking teaching methods used in the selected works, including the HCI evaluation techniques of these tools, and a discussion to answer the research questions proposed in section 3. Section 5 discusses the threats to study validity. Finally, section 6 presents the final considerations of this paper.

2 Related Works

This section focuses on other systematic mappings/reviews of the literature published over the past years by other researchers. In contrast, this paper stands out for being a more recent mapping (2014 to 2024) and for bringing the evaluation methods used for the tools found based on HCI strategies.

The systematic literature review conducted by Robe et al. (2020) aimed to identify scientific studies presenting the categories of pedagogical resources adopted in programming education for visually impaired students. The authors identified five categories of pedagogical resources: (1) programmable devices combined with other resources; (2) physical programming languages (PPL); (3) block-based programming environments (BBPEs); (4) accessible programming environments; (5) haptic resources. Furthermore, challenges were identified, such as difficulties in navigating complex codes with screen readers, the lack of tools for collaboration and code sharing between visually impaired individuals, and the need to integrate pedagogical resources with auditory and tactile feedback.

Hadwen-Bennett et al. (2018) conducted a literature review on how to make programming teaching accessible for visually impaired students, evaluating different strategies used for this purpose and identifying areas that require further research. The review classified the approaches into four main themes: (1) making text-based programming languages accessible; (2) making block-based programming languages accessible; (3) use of physical artifacts; and (4) auditory and haptic feedback.

Oliveira et al. (2019) conducted a systematic literature review on the use of robotics in programming education for visually impaired individuals, covering the period from 2016 to 2019. The main goal was to identify existing studies, the methodologies used, the robotics and programming kits, and good practices and challenges faced in teaching programming with robotic support for this population. The review pointed out that many current programming tools and robotic environments are inaccessible to visually impaired students due to the predominant use of graphical interfaces.

The article by Al-Ratta et al. (2013) is a systematic literature review of articles published between 1975 and 2013 on programming education for blind individuals. The main objective was to provide an overview of academic research on the topic, categorizing it by areas of interest and conducting a quantitative analysis based on type and year of publication. The article also explored future research directions, identifying gaps in the reviewed studies. The work aimed to highlight the specific needs of blind individuals to learn programming and discuss whether there is a need to develop specific programming languages for this population.

3 The Systematic Literature Mapping

This paper was constructed based on a systematic literature mapping, an appropriate method for mapping a specific topic when there is limited evidence or the research topic is broad or dispersed. Petersen et al. (2015) highlight that systematic mapping studies structure a research area, while systematic reviews focus on gathering and synthesizing evidence. In this paper, the guidelines proposed by Kitchenham (2004) were followed to conduct the study. Firstly, primary works were sought to synthesize the methods of teaching programming logic to visually impaired individuals and identify evaluation methods in HCI and gaps that have yet to be explored by researchers. Furthermore, this study presented five research questions, including selection, classification, and data extraction protocols. The mapping was conducted by selecting studies within eleven years, from January 2014 to December 2024.

3.1 Research Questions

In this work, five research questions were defined to be answered to discover which algorithm teaching tools have been used by researchers and which HCI evaluation techniques have been conducted.

RQ1. What are the studies' primary target audiences and their main characteristics? Here, the goal is to understand the group the work targets. This can include age range (children, young people, adults, elderly), educational background (elementary school, high school, undergraduate), or a combination of these groups.

RQ2. What are the main difficulties people report while learning programming logic? It is important first to understand the limitations faced by these individuals during the learning process and how they react to these challenges.

RQ3. What teaching tools for programming logic have been used? The aim here is to identify which methods, techniques, and tools have been used by educators and researchers to assist in teaching algorithms to people with visual disabilities.

RQ4. In which contexts have these programming learning support tools for PwVI been used? The idea is to understand the use context in which the method was applied.

RQ5. Have these tools been evaluated? If so, have HCI evaluation methods been adopted for these tools? Finally, the aim is to understand the evaluation methods used for these teaching tools and whether any HCI techniques were employed to analyze the tools used.

3.2 Research Conduction

To conduct the systematic literature mapping, an extensive bibliographic review was first carried out to identify relevant keywords related to the topic in question. Then, three databases were selected: ACM Digital Library, IEEE Xplore, and Springer. The choice reflects these databases' relevance for the computing field, including various published studies on computing education. A specific search string was also used, incorporating terms related to programming logic and including visually impaired individuals. The search string created and applied was:

(mentoring OR teaching) AND (programming OR "logical reasoning") AND ("blind people" OR "visual impairment")

Table 1 presents the inclusion and exclusion criteria applied in the research. Two inclusion criteria were used to select the articles that would answer the research questions: (i) the article must focus on teaching methodologies for programming logic for visually impaired individuals; (ii) the method used in the article was tested and verified using HCI evaluation techniques. The search result for articles in the three databases returned 1466 articles, with 955 from the ACM Digital Library, 40 from IEEE Xplore, and 471 from Springer.

The careful analysis of the resulting articles focused on pedagogical strategies, assistive technologies, and inclusive practices applied in the context of teaching programming logic. This specific methodological approach allowed for a deeper understanding of the academic contributions during the established period, guiding educators, researchers, and professionals in advancing digital inclusion in computer science education for visually impaired students.

Table 1: Paper Selection Criteria

	Inclusion Criteria
*IC*1:	The article must focus on teaching methodologies for programming logic for visually impaired individuals.
*IC*2:	The method used in the article was tested and verified using HCI evaluation techniques.
	Exclusion Criteria
*EC*1:	Lectures, tutorials, and any non-peer-reviewed studies;
*EC*2:	Duplicated papers;
*EC*3:	Duplicate studies (the most comprehensive article should be selected);
*EC*4:	Secondary studies;
*EC*5:	Studies where the full text is not in Portuguese or English;
*EC*6:	Studies that were not related to the research questions (mainly RQ3 and RQ5);

After applying the inclusion and exclusion criteria, the number of selected studies was 13 articles. Some studies presented methods for teaching programming logic to visually impaired individuals, but did not mention or apply any evaluation techniques for the methods used. Figure 1 shows the stages of the article selection process.

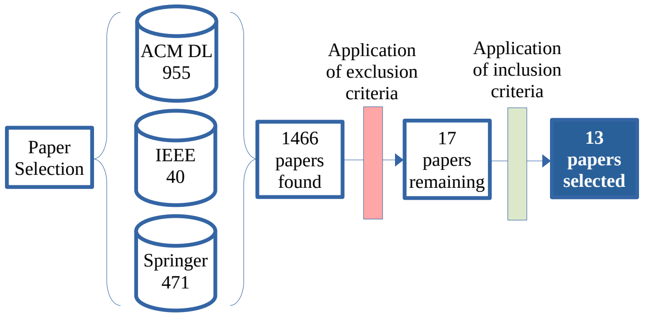

Figure 1: Selection and Extraction of Studies. Source: The Authors.

4 Results and discussion

This section will discuss the analyses and reflections regarding the results obtained from implementing the previous phases of the selected research methodology. The selected articles can be accessed through this link[1]. The five questions that guided our investigation will be answered below:

RQ1. What are the studies' primary target audiences and their main characteristics?

Among the selected articles, some works focus on teaching logic programming to children. The research by Thieme et al. (2017) is aimed at children with and without visual impairments, mainly aged between 7 and 11 years old. Pires et al. (2020) focus on children with visual impairments, emphasizing ages 5 to 11. Utreras et al. (2020) also focus on blind and low-vision children and children without visual impairments. Finally, Barbareschi et al. (2020) define the target audience as children with mixed visual abilities aged 6 to 11.

Furthermore, other articles target a younger audience. The work by Papazafiropulos et al. (2016) focuses on students with and without visual impairments, exceptionally high school students. Ludi et al. (2014) direct the study toward blind and low-vision adolescents participating in educational robotics programs. Kakehashi et al. (2014) target individuals with visual impairments, specifically elementary and middle school students.

Some articles do not focus on a specific age range but target people with or without visual impairments. The research by Nascimento et al. (2023) targets individuals with visual impairments, particularly blind users wanting to learn programming. Pereira et al. (2018) focus on students with visual impairments (total blindness and low vision). Rong et al. (2020) direct the study to students with visual impairments and sighted students. Oliveira et al. (2020) focus on individuals with visual impairments, who may be either beginners or experienced in programming. Konecki et al. (2016) target students with visual impairments and those taking introductory programming courses, as well as students with comorbidities such as Attention Deficit Hyperactivity Disorder (ADHD) and Global Developmental Delay (GDD). Finally, Lotlikar et al. (2020) focus on students with visual impairments.

Based on the analyzed articles, it is evident that most studies focus on children and adolescents, especially during the school period, with approaches aimed at teaching programming to students

[1] https://bit.ly/4aG48hP

with and without visual impairments. The predominant age groups are between 5 to 11 years old and high school students, with a significant emphasis on blind or low-vision children.

Despite studies that do not define specific age ranges, targeting people with visual impairments in general, there is a noticeable absence of research focusing on higher education. This gap indicates that the challenge of including visually impaired undergraduates in advanced programming courses and related fields remains underexplored, requiring greater attention from the academic community to expand inclusion and accessibility strategies in more complex levels of education.

RQ2. What are the main difficulties people report while learning programming logic?

Blind or low-vision individuals face several barriers to accessing programming curricula, such as the difficulty screen readers have in dealing with complex graphical interfaces, which hinders access to modern development environments for students with visual impairments [25], the lack of accessibility in learning materials, and inaccessible coding editors and environments, such as Scratch[2], Blockly[3], and Alice[4], which are widely used in teaching computational thinking but present accessibility barriers, especially for people with visual impairments [11]. According to Ludi et al. (2014) and Kakehashi et al. (2014), most current programming environments are highly graphical and thus inaccessible for individuals with visual impairments. They mention these individuals' difficulties, such as the lack of audio correspondence for graphical information and incompatibility with assistive technologies.

There is also a disconnect between teaching methods and students' needs, as they may have different learning styles and require personalized methodologies [15]. Another recurring problem is the excessive focus on syntax in general (Python, C++, Java, HTML, etc), as many courses focus on programming language syntax rather than problem-solving, which, according to Bergin et al. (1999) and Kölling (2001), can be a flawed approach .

The analyzed articles converge on reflections about the lack of accessible resources for programming education. Blockly and Blocks4All attempt to create accessible versions of visual programming systems but face difficulties related to reading blocks and screen reader navigation [31, 33]. The standard software used to program Lego Mindstorms robots uses icons and is inaccessible to screen readers, making it difficult for blind students to use [7]. In Kakehashi et al. (2014), the paper cites other studies and tools that use tangible interfaces, such as E-Block, Tern, and AlgoBlock, which focus on teaching programming concepts to beginners. However, many of these applications heavily depend on visual elements, such as buttons and line locators, making it significantly difficult for visually impaired individuals to access the programming world.

The challenges faced by blind or low-vision individuals in accessing programming education are vast and still largely unresolved. The predominance of graphical interfaces in modern development environments, coupled with the lack of accessibility to educational materials and tools such as Scratch, Blockly, and Alice, reveals a significant gap in technological inclusion. Although initiatives such as Blockly and Blocks4All strive to create more accessible solutions, their practical limitations, such as difficulties in navigation by screen readers, highlight how insufficient accessibility still is. Furthermore, the excessive focus on syntax and the disconnect between teaching methods and students' needs only exacerbate the problem. Thus, it is evident that accessible resources for teaching programming are scarce.

RQ3. What teaching tools for programming logic have been used?

Nascimento et al. (2023) present the IVProg4All tool, which utilizes a visual programming system based on HTML/CSS to foster interaction between students with and without visual impairments, creating an inclusive learning environment. In the article by Pereira et al. (2018), the use of physical flowcharts and screen readers (such as DOSVOX, JAWS, and NVDA) combined with the Pascal programming language represents a tactile and auditory approach to understanding algorithms. This duality of methods allows students with and without visual impairments to participate equally. In Ludi et al. (2014), JBrick, designed for Lego Mindstorms NXT, is an accessible tool that enables visually impaired students to program robots in a collaborative environment using screen readers and Braille displays. These approaches facilitate collaboration between visually impaired and sighted students, engaging them in shared learning experiences and promoting a more interactive and flexible educational environment.

Thieme et al. (2017) introduce the Torino language, which combines physical pieces called "beads" with the cooperative inquiry method to teach programming to children aged 7 to 11, including those with visual impairments. Manipulating physical beads to create programs is a central feature of this tool. In the work of Rong et al. (2020), the CodeRhythm tool is presented as a set of tangible blocks that teach programming through melodies. The blocks, representing musical notes and control functions, provide auditory and tactile feedback, making the experience accessible for visually impaired individuals. Kakehashi et al. (2014) present the P-CUBE system, which uses programmable blocks with RFID to teach programming. Visually impaired students controlled a mobile robot through sequential programming, loops, and conditionals. In the article by Utreras et al. (2020), a tangible programming prototype using Lego blocks and an Arduino Mega microcontroller was tested with visually impaired adults, enabling the representation of programming instructions through a physical interface. Barbareschi et al. (2020) introduced TIP-Toy, an open-source kit combining physical blocks to teach basic programming concepts, promoting fun and interaction among children with mixed visual abilities. All these tools utilize physical components to facilitate learning. This tactile approach is essential for including visually impaired students, allowing them to manipulate tangible objects to understand abstract programming concepts.

In the research by Oliveira et al. (2020), the Donnie programming environment, featuring the GoDonnie language, was developed to teach robot programming in virtual environments, focusing on developing orientation and mobility skills alongside computational thinking. Pires et al. (2020) present a study on tangible programming involving creating an educational environment where visually impaired children learn programming using physical blocks and robots. Practical activities and group discussions were crucial in

[2]https://scratch.mit.edu/about
[3]https://developers.google.com/blockly
[4]https://www.alice.org/

understanding the students' needs. In Papazafiropulos et al. (2016), tactile models for teaching sorting algorithms, using 3D printing, allowed students to manipulate physical data representations, facilitating understanding through touch. In Konecki et al. (2016), the Audio-Based Programming Tutor (ABPT) system is a voice-controlled tool that offers audio lessons and virtual assistance, enabling visually impaired students to learn programming without relying on visual interfaces. Finally, in the article by Lotlikar et al. (2020), a pilot study using tangible flowchart blocks was conducted to develop logical thinking in visually impaired students. The hands-on approach allowed participants to apply programming concepts to everyday problems.

The analyzed tools reflect a diversity of innovative approaches to teaching programming logic to visually impaired individuals. Many employ tangible elements, such as physical blocks and robotics kits, which transform abstract concepts into practical and tactile experiences, essential for promoting inclusion and understanding. Others invest in accessible technologies such as screen readers, audio-based systems, and adapted programming environments, fostering collaborative interaction between visually impaired and sighted students. However, despite the progress represented by these initiatives, it is evident that many of them still focus on the early stages of education, indicating the need to expand these tools to reach more advanced and diverse audiences. Thus, the studies suggest that combining physical, auditory, and digital resources has the potential to overcome accessibility barriers but requires more significant investment to become more widely available and applicable in different educational contexts.

RQ4. In which contexts have these programming learning support tools for PwVI been used?
Based on the scenarios observed in the articles, we can identify interrelations among them regarding usage contexts, pedagogical objectives, and inclusion for visually impaired individuals in programming education.

Collaboration Between Visually Impaired and Sighted Students. IVProg4All [11], Torino [44], CodeRhythm [41], 3D Haptic Models [34], Tangible Programming with Lego Blocks [45], and TIP-Toy [3]: These tools focus on creating inclusive learning environments where visually impaired and sighted students can learn together. For example, IVProg4All enables collaboration between blind and sighted individuals, just as CodeRhythm aims to include both groups. 3D models and tangible interfaces facilitate collaborative teaching, allowing students to learn the same subject but through different approaches tailored to their visual needs.

Use of Tangible Tools and Physical Components. Torino [44], Physical Flowcharts and Screen Readers [35], CodeRhythm [41], Programming Blocks and Robots [37], 3D Haptic Models [34], Tangible Flowchart Blocks [28], P-CUBE [20], Tangible Programming with Lego Blocks [45], and TIP-Toy [3]: These tools employ physical blocks or tangible elements, such as beads, blocks, or robots, to teach programming. In the case of Torino, physical beads allow children to create programs by manipulating these components. Similarly, P-CUBE and TIP-Toy use physical blocks to promote interaction with programming concepts. This approach helps overcome reliance on digital visual interfaces, significantly benefiting visually impaired students.

Application in Robotics and Virtual Environments. Donnie / GoDonnie [9], Programming Blocks and Robots [37], JBrick [29], and P-CUBE [20]: These tools focus on teaching programming in a robotics context, providing a more practical and applied learning experience. Tools like JBrick and P-CUBE concentrate on giving students the experience of programming robots. At the same time, the Donnie environment supports programming education and aids in developing Orientation and Mobility (O&M) skills through interaction with robots in a virtual environment.

Alternatives to Traditional Software Development Systems. IVProg4All [11], Donnie / GoDonnie [9], Audio-Based Programming Tutor [25], and JBrick [29]: The Audio-Based Programming Tutor (ABPT), for example, focuses on overcoming the barriers visually impaired individuals face in using software development tools, which often rely heavily on graphical interfaces. It provides an alternative for blind students to learn and use typically inaccessible tools, addressing a critical accessibility issue in professional programming contexts.

These tools form an ecosystem of inclusive pedagogical solutions that combine tangible interfaces, accessibility, and robotics to teach programming to visually impaired individuals. Some focus on creating collaborative environments for blind and sighted students, others offer physical and tangible interfaces that bypass reliance on visual interfaces, and many integrate robotics as a means to make learning practical and enjoyable.

RQ5. Have these tools been evaluated? If so, have HCI evaluation methods been adopted for these tools?
The HCI evaluation techniques used in teaching programming logic to visually impaired individuals (PwVI) are listed in Table 2. The most commonly used methods were user testing, questionnaires, and interviews. Figure 2 presents the distribution of HCI evaluations across the articles.

The Cognitive Walkthrough [38] is an incremental learning style that allows a balanced and necessary effort to master a new function, partly due to the immediate value that this function offers to the user.

User testing, also known as usability testing, evaluates the ease of use of interactive systems based on the direct experience of target users [42].

The "think-aloud" method investigates mental processes, asking participants to verbalize their thoughts while performing a task [26].

A questionnaire, according to Gil (2008), is defined as an investigation technique consisting of a varied set of written questions directed at participants, aiming to gather information about their opinions, beliefs, feelings, interests, expectations, experiences, among other aspects.

According to Haguete (2001) , the interview values the use of words as a symbol in human relationships, allowing participants to build and understand the reality around them, as argued by Jovechlovitch and Bauer (2002) .

Focus groups are a qualitative research technique that collects information through group interactions, as defined by Morgan (1997) and Kitzinger (1995) .

5 Threats to Study Validity

Primary Study Selection. To ensure impartiality in the selection process, research questions were established, and inclusion and

Table 2: HCI Evaluation Techniques Used in the Selected Articles.

Method	Evaluation
IVProg4All [11]	User Testing, Cognitive Walkthrough and Think-Aloud
Torino [44]	User Testing
Physical Flowchartss e Screen Readers [35]	Questionnaire
CodeRhythm [41]	User Testing
Donnie / GoDonnie [9]	User Testing, Questionnaire and Interview
Programming Blocks and Robots [37]	Focus Groups and User Testing
3D Haptic Models [34]	User Testing
Audio Based Programming Tutor (ABPT) [25]	User Testing and Questionnaire
JBrick [29]	Field Studies, Questionnaire and Interview
Tangible Flowchart Blocks [28]	User Testing and Interview
P-CUBE [20]	User Testing and Questionnaire
Tangible Programming with Lego Blocks [45]	User Testing, Questionnaire and Interview
TIP-Toy [3]	User Testing

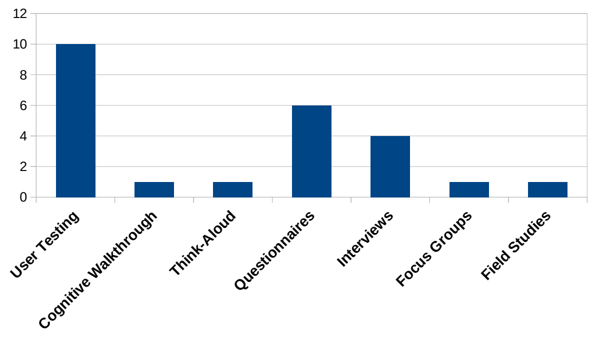

Figure 2: Distribution of HCI Evaluations Across Papers.

exclusion criteria were defined to identify relevant studies. Furthermore, the study screening was conducted independently by two researchers, who later compared their results. In cases of disagreement regarding the inclusion of an article, the researchers discussed to reach an agreement. If consensus could not be reached, the final decision was made by a third researcher.

Relevant Primary Studies Not Included. Although three sources were used to select the primary studies, it is possible that some were inadvertently overlooked. To mitigate this limitation, we chose sources that index studies from major scientific databases related to "inclusive education in computing" and cover most of the relevant articles on the topic.

Interpretative Validity. Interpretative validity is ensured when the conclusions drawn are reasonable given the data, thus reflecting the validity of the conclusion [36]. One of the major risks in data interpretation is researcher bias, which was minimized by having two co-authors review the study: a professor with a PhD and an undergraduate student.

Repeatability. In this research, several steps were implemented to maximize repeatability, including the clear definition of inclusion and exclusion criteria, providing the list of selected articles along with the criteria used for inclusion, and utilizing a structured data extraction form.

6 Conclusions

The teaching tools identified in this research and their respective HCI evaluation methods aim to improve the experience of visually impaired users in learning programming logic, providing a more accessible and motivating alternative than traditional methods. This study enabled the identification of educational tools designed for visually impaired individuals and their respective HCI evaluations using 13 articles collected from three relevant academic databases.

It is important to highlight that, in most cases, the evaluation of the educational tools occurred after their implementation. In contrast, assessments conducted during the development process could ensure more effective results. Among the works identified in the systematic mapping, few effectively addressed which aspects of the tools should be improved or suggested solutions to the problems found, limiting themselves to HCI evaluation.

As future work, it is suggested that studies be conducted proposing a unified and more suitable method for evaluating educational tools aimed at teaching programming logic to visually impaired individuals, particularly in terms of their effectiveness in HCI.

References

[1] Nusaibah M. Al-Ratta and Hend S. Al-Khalifa. 2013. Teaching programming for blinds: A review. In *Fourth International Conference on Information and Communication Technology and Accessibility (ICTA).* 1–5. doi:10.1109/ICTA.2013.6815285

[2] João Arruda, Pedro Henrique Rocha, Regiane Francês, and Victor Hugo Pinto. 2024. Explorando a Robótica para mitigar Desafios Comportamentais e de Aprendizado em Programação na Graduação. In *Anais do XXX Workshop de Informática na Escola* (Rio de Janeiro/RJ). SBC, Porto Alegre, RS, Brasil, 243–253. doi:10.5753/wie.2024.242728

[3] Giulia Barbareschi, Enrico Costanza, and Catherine Holloway. 2020. TIP-Toy: a tactile, open-source computational toolkit to support learning across visual abilities. In *Proceedings of the 22nd International ACM SIGACCESS Conference on Computers and Accessibility* (Virtual Event, Greece) *(ASSETS '20).* Association for Computing Machinery, New York, NY, USA, Article 21, 14 pages. doi:10.1145/3373625.3417005

[4] Joseph Bergin, Viera K. Proulx, Alyce Faulstich Brady, Stephen Hartley, Charles Kelemen, Frank Klassner, Amruth Kumar, Myles McNally, David Mutchler, Richard Rasala, and Rocky Ross. 2023. Resources for next generation introductory CS courses: report of the ITiCSE'99 working group on resources for the next generation CS 1 course. *SIGCSE Bull.* 31, 4 (Dec. 2023), 101–105. doi:10.1145/349522.349555

[5] João Batista Bottentuit Junior and Clara Pereira Coutinho. 2009. Podcast : uma ferramenta tecnológica para auxílio ao ensino de deficientes visuais.

[6] Stacy M. Branham and Shaun K. Kane. 2015. Collaborative Accessibility: How Blind and Sighted Companions Co-Create Accessible Home Spaces. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15).* Association for Computing Machinery, New York, NY, USA, 2373–2382. doi:10.1145/2702123.2702511

[7] Kelly R. Cannon, Katherine A. Panciera, and Nikolaos P. Papanikolopoulos. 2007. Second annual robotics summer camp for underrepresented students. *SIGCSE Bull.* 39, 3 (June 2007), 14–18. doi:10.1145/1269900.1268791

[8] Dino Capovilla, Johannes Krugel, and Peter Hubwieser. 2013. Teaching Algorithmic Thinking Using Haptic Models for Visually Impaired Students. In *Proceedings of the 2013 Learning and Teaching in Computing and Engineering (LATICE '13).* IEEE Computer Society, USA, 167–171. doi:10.1109/LaTiCE.2013.14

[9] Juliana Damasio Oliveira, Márcia de Borba Campos, and Vanessa Stangherlin Machado Paixão-Cortes. 2020. Usable and Accessible Robot Programming System for People Who Are Visually Impaired. In *Universal Access in Human-Computer Interaction. Design Approaches and Supporting Technologies*, Margherita Antona

and Constantine Stephanidis (Eds.). Springer International Publishing, Cham, 445–464.

[10] Elizabet Dias de Sá, Izilda Maria de Campos, and Myriam Beatriz Campolina Silva. 2007. *Formação continuada a distância de professores para o atendimento educacional especializado - Deficiência visual.* Retrieved August 25, 2024 from https://observatoriodeeducacao.institutounibanco.org.br/api/assets/observatorio/e20c71b2-19e0-49f5-bad1-278c0323fa63/

[11] Marcos D. Do Nascimento, Anarosa A. F. Brandão, Leônidas De Oliveira Brandão, and Tiago Melo Casal. 2023. Towards iVProg4All: An Accessibility Test with Blind. In *2023 IEEE Frontiers in Education Conference (FIE).* 01–05. doi:10.1109/FIE58773.2023.10343224

[12] Juliana Damasio e Darlan Jurak e Robson Bittencourt e Márcia Campos e Alexandre Amory. 2019. Programming teaching with robotic support for people who are visually impaired: a systematic review. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)* 30, 1 (2019), 1231. doi:10.5753/cbie.sbie.2019.1231

[13] Caique Ferreira e João Anjos e Joao Normando e Milton Castro e Valguima Odakura e Rodrigo Sacchi e Carla Barvinski. 2016. Uso de podcast para apoio a aprendizagem de algoritmos em curso de graduação em Computação. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação* 5, 1 (2016), 1208. doi:10.5753/cbie.wcbie.2016.1208

[14] Antonio Carlos Gil. 2008. *Métodos e técnicas de pesquisa social.* 6. ed. Editora Atlas SA.

[15] Anabela Gomes and António José Mendes. 2007. Learning to program-difficulties and solutions. In *International Conference on Engineering Education–ICEE*, Vol. 7. 1–5.

[16] Mark Guzdial, Alan Kay, Cathie Norris, and Elliot Soloway. 2019. Computational thinking should just be good thinking. *Commun. ACM* 62, 11 (Oct. 2019), 28–30. doi:10.1145/3363181

[17] Alex Hadwen-Bennett, Sue Sentance, and Cecily Morrison. 2018. Making Programming Accessible to Learners with Visual Impairments: A Literature Review. *International Journal of Computer Science Education in Schools* 2, 2 (May 2018), 3–13. doi:10.21585/ijcses.v2i2.25

[18] T.M.F. Haguette. 2001. *Metodologias qualitativas na sociologia.* Vozes. https://books.google.com.br/books?id=eteVSAAACAAJ

[19] Sandra Jovchelovitch and Martin W Bauer. 2002. Entrevista narrativa. *Pesquisa qualitativa com texto, imagem e som: um manual prático* 4 (2002), 90–113.

[20] Shun Kakehashi, Tatsuo Motoyoshi, Ken'ichi Koyanagi, Toru Oshima, Hiroyuki Masuta, and Hiroshi Kawakami. 2014. Improvement of P-CUBE: Algorithm education tool for visually impaired persons. In *2014 IEEE Symposium on Robotic Intelligence in Informationally Structured Space (RiiSS).* 1–6. doi:10.1109/RIISS.2014.7009180

[21] Barbara Kitchenham. 2004. Procedures for performing systematic reviews. (2004).

[22] Jenny Kitzinger. 1995. Qualitative Research: Introducing focus groups. *BMJ* 311, 7000 (1995), 299–302. doi:10.1136/bmj.311.7000.299 arXiv:https://www.bmj.com/content

[23] Cristian Koliver, Ricardo Vargas Dorneles, and Marcos Eduardo Casa. 2004. Das (muitas) dúvidas e (poucas) certezas do ensino de algoritmos. *Anais do XII Workshop de Educação em Computação* 24 (2004).

[24] Michael Kölling and John Rosenberg. 2001. Guidelines for teaching object orientation with Java. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education* (Canterbury, United Kingdom) *(ITiCSE '01).* Association for Computing Machinery, New York, NY, USA, 33–36. doi:10.1145/377435.377461

[25] Mario Konecki, Nikola Ivković, and Matija Kaniški. 2016. Making programming education more accessible for visually impaired. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).* 887–890. doi:10.1109/MIPRO.2016.7522265

[26] C. Lewis. [n. d.]. *Using the "thinking Aloud" Method in Cognitive Interface Design.* IBM Thomas J. Watson Research Division. https://books.google.com.br/books?id=F5AKHQAACAAJ

[27] Dastyni Loksa, Amy J. Ko, Will Jernigan, Alannah Oleson, Christopher J. Mendez, and Margaret M. Burnett. 2016. Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16).* Association for Computing Machinery, New York, NY, USA, 1449–1461. doi:10.1145/2858036.2858252

[28] Priya Lotlikar, Deepak Pathak, P. C. Herold, and chandan Dasgupta. 2020. Tangible Flowchart Blocks for Fostering Logical Thinking in Visually Impaired Learners. In *2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT).* 266–268. doi:10.1109/ICALT49669.2020.00087

[29] Stephanie Ludi, Lindsey Ellis, and Scott Jordan. 2014. An accessible robotics programming environment for visually impaired users. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility* (Rochester, New York, USA) *(ASSETS '14).* Association for Computing Machinery, New York, NY, USA, 237–238. doi:10.1145/2661334.2661385

[30] Craig M. MacDonald and Michael E. Atwood. 2013. Changing perspectives on evaluation in HCI: past, present, and future. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems* (Paris, France) *(CHI EA '13).* Association for Computing Machinery, New York, NY, USA, 1969–1978. doi:10.1145/2468356.2468714

[31] Lauren R. Milne and Richard E. Ladner. 2018. Blocks4All: Overcoming Accessibility Barriers to Blocks Programming for Children with Visual Impairments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18).* Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3173574.3173643

[32] D.L. Morgan. 1997. *Focus Groups as Qualitative Research.* SAGE Publications. https://books.google.com.br/books?id=iBJZusd1GocC

[33] Aboubakar Mountapmbeme, Obianuju Okafor, and Stephanie Ludi. 2022. Accessible Blockly: An Accessible Block-Based Programming Library for People with Visual Impairments. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility* (Athens, Greece) *(ASSETS '22).* Association for Computing Machinery, New York, NY, USA, Article 19, 15 pages. doi:10.1145/3517428.3544806

[34] Nicola Papazafiropulos, Luca Fanucci, Barbara Leporini, Susanna Pelagatti, and Roberto Roncella. 2016. Haptic Models of Arrays Through 3D Printing for Computer Science Education. In *Computers Helping People with Special Needs*, Klaus Miesenberger, Christian Bühler, and Petr Penaz (Eds.). Springer International Publishing, Cham, 491–498.

[35] Rodolfo M. Pereira, Felippe Fernandes da Silva, and Carlos N. Silla. 2018. Teaching Algorithms for Visually Impaired and Blind Students using Physical Flowcharts and Screen Readers. In *2018 IEEE Frontiers in Education Conference (FIE).* 1–9. doi:10.1109/FIE.2018.8658511

[36] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18. doi:10.1016/j.infsof.2015.03.007

[37] Ana Cristina Pires, Filipa Rocha, Antonio José de Barros Neto, Hugo Simão, Hugo Nicolau, and Tiago Guerreiro. 2020. Exploring accessible programming with educators and visually impaired children. In *Proceedings of the Interaction Design and Children Conference* (London, United Kingdom) *(IDC '20).* Association for Computing Machinery, New York, NY, USA, 148–160. doi:10.1145/3392063.3394437

[38] Peter G. Polson, Clayton Lewis, John Rieman, and Cathleen Wharton. 1992. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies* 36, 5 (1992), 741–773. doi:10.1016/0020-7373(92)90039-N

[39] T. V. Raman. 1996. Emacspeak—a speech interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, British Columbia, Canada) *(CHI '96).* Association for Computing Machinery, New York, NY, USA, 66–71. doi:10.1145/238386.238405

[40] Rafaela Robe, Bruna Poletto Salton, and Silvia Bertagnolli. 2020. RECURSOS PEDAGÓGICOS PARA O ENSINO DE PROGRAMAÇÃO DE ESTUDANTES COM DEFICIÊNCIA VISUAL: UMA REVISÃO SISTEMÁTICA DA LITERATURA. *Revista Novas Tecnologias na Educação* 18, 1 (jul. 2020). doi:10.22456/1679-1916.105922

[41] Zhiyi Rong, Ngo fung Chan, Taizhou Chen, and Kening Zhu. 2020. CodeRhythm: A Tangible Programming Toolkit for Visually Impaired Students. In *Proceedings of the Eighth International Workshop of Chinese CHI* (Honolulu, HI, USA) *(Chinese CHI '20).* Association for Computing Machinery, New York, NY, USA, 57–60. doi:10.1145/3403676.3403683

[42] J. Rubin. 1994. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests.* Wiley. https://books.google.com.br/books?id=j-BQzDVsUGcC

[43] Andreas Stefik, Andrew Haywood, Shahzada Mansoor, Brock Dunda, and Daniel Garcia. 2009. SODBeans. In *2009 IEEE 17th International Conference on Program Comprehension.* 293–294. doi:10.1109/ICPC.2009.5090064

[44] Anja Thieme, Cecily Morrison, Nicolas Villar, Martin Grayson, and Siân Lindley. 2017. Enabling Collaboration in Learning Computer Programing Inclusive of Children with Vision Impairments. In *Proceedings of the 2017 Conference on Designing Interactive Systems* (Edinburgh, United Kingdom) *(DIS '17).* Association for Computing Machinery, New York, NY, USA, 739–752. doi:10.1145/3064663.3064689

[45] Emmanuel Utreras and Enrico Pontelli. 2020. Design of a Tangible Programming Tool for Students with Visual Impairments and Low Vision. In *Universal Access in Human-Computer Interaction. Applications and Practice*, Margherita Antona and Constantine Stephanidis (Eds.). Springer International Publishing, Cham, 304–314.

[46] Greg Wilson. 2017. *How to Teach Programming (And Other Things).* Lulu.com.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009