Accessible Representations of Visual Artifacts in Technical Informatics Education

Diethelm Bienhaus, Michael Kreutzer, Florian von Zabiensky
Institute of Technique and Informatics, Department of Mathematics, Natural Sciences and Computer Science
University of Applied Sciences Mittelhessen
Gießen, Germany
e-mail: {diethelm.bienhaus | michae.kreutzer | florian.von.zabiensky}@mni.thm.de

***Abstract*—Graphical representations and diagrams are widely used in engineering disciplines. In computer science and electrical engineering, Unified Modeling Language (UML) diagrams, electronic circuit schematics, and Karnaugh-Veitch (KV) diagrams are commonplace. At our university, they are taught in courses of Technical Informatics and Software Engineering. However, these visual methods pose considerable barriers for students with visual impairments. Hence, using graphical representations is limiting educational accessibility and inclusivity. In response, this paper presents text-based alternatives which we have been using since three semesters to provide equivalent conceptual information in non-visual ways.**

Keywords-Accessibility; Visual Impairment; Inclusive STEM Education; UML Class Diagrams; Electronic Circuits Simulation; Logic Gate Modeling; Textual Representations; Educational Technology; Screen Reader Compatibility.

I. Introduction

Visual representations, including Unified Modeling Language (UML) diagrams, electronic circuit schematics, logic gates, and Karnaugh-Veitch (KV) diagrams, constitute foundational elements in technical informatics and software engineering education. These tools are employed universally to convey complex abstract and concrete concepts efficiently. Despite their educational value, these graphical representations pose significant accessibility challenges, rendering them unsuitable for blind and visually impaired students. Consequently, there is a pressing imperative to develop and implement effective alternatives to ensure equitable access to technical education, in line with international frameworks, such as the United Nations Convention on the Rights of Persons with Disabilities (UN-CRPD).

Graphical diagrams inherently rely on visual-spatial reasoning, which is inaccessible without visual perception or assistive adaptations. For visually impaired learners, these diagrams represent barriers that limit participation, understanding, and ultimately educational achievement in technical disciplines. This exclusion not only hinders individual educational outcomes but also contravenes principles of inclusivity and equal opportunity embedded in modern educational policies.

The motivation behind the present study emerged from practical challenges encountered at the University of Applied Sciences Mittelhessen (Technische Hochschule Mittelhessen: THM), where visually impaired students enrolled in technical informatics courses. Their participation highlighted an urgent need to adapt existing pedagogical materials to accessible formats. Hence, we initiated a focused research and development effort to address these challenges systematically.

Specifically, this paper addresses the following research questions: How can standard graphical representations in technical informatics be effectively translated into accessible text-based formats? What approaches ensure semantic equivalence and pedagogical effectiveness for both visually impaired and sighted students? Our primary objectives are to develop robust, text-based alternatives to conventional diagrams and evaluate their educational viability and inclusivity through empirical validation.

The remainder of this paper is structured as follows: Section II provides a critical review of existing approaches to accessible diagram representation. Section III presents our developed text-based solutions, detailing their conceptualization, implementation, and deployment. In Section IV, we describe our evaluation methodology and present the findings from structured interviews with visually impaired students. Finally, Section V discusses implications, limitations, and avenues for future research, concluding with a summary of our contributions.

The motivation for developing the solutions presented here stemmed from the enrollment of blind students in computer science at THM. This posed the challenge of how graphical representations, such as UML class diagrams, electronic circuit diagrams, and logic circuits could be supplemented or replaced by purely textual representations.

II. Related Work

Ensuring accessibility of visual diagrams in STEM education has increasingly gained attention due to the imperative of providing equal learning opportunities for visually impaired students.

In the following subsections, we analyze major existing approaches, evaluating their potential and limitations in relation to our work.

A. Multimodal and Universal Design Approaches

Multimodal systems employing vibro-audio interfaces offer visually impaired students alternative interaction methods with visual content. Doore et al. [1] demonstrated that these systems effectively support both visually impaired and sighted students, underscoring the merits of universal design in educational environments. However, such solutions often require specialized hardware and considerable developmental effort, potentially limiting their scalability within typical educational settings.

B. Text-based Approaches for UML and Circuits

Text-based alternatives provide a pragmatic, cost-effective solution. Wildhaber et al. [2] proposed a method enabling visually impaired students to independently create and edit UML diagrams using accessible mobile interfaces. Their approach lowers the entry barrier compared to conventional textual UML tools like PlantUML and YAML. This work aligns with our experiences, confirming that textual representations not only mitigate accessibility barriers but also offer pedagogical benefits by emphasizing semantic structuring over visual complexity.

C. Tactile and Haptic Representations

Tactile approaches, such as the thermo-formed paper method proposed by Pissaloux et al. [3] for representing complex diagrams, facilitate haptic access to visual information. While these solutions are valuable, their creation and distribution are costly and logistically complex. Additionally, they lack flexibility and real-time adaptability, posing disadvantages in dynamic teaching contexts.

D. Interactive and Tangible Representations

Studies by Ducasse et al. [4][5] on interactive and tangible maps have highlighted the benefits of physical and digital interaction for spatial understanding among visually impaired individuals. Although their focus was on geographic content, the underlying principles of interactivity and tactile feedback could potentially transfer to other visual domains like electronic circuits. Yet, these approaches similarly demand specialized technology and significant spatial resources.

E. Synthesis and Connection to Our Work

The reviewed works illustrate diverse strategies for improving visual diagram accessibility. While multimodal and haptic methods constitute valuable additions, their inherent limitations include substantial resource demands and scalability issues. In contrast, text-based approaches such as PlantUML and YAML descriptions provide feasible, scalable alternatives. Our work builds directly upon these textual solutions, further developing them to offer semantically equivalent, pedagogically sound, and accessible alternatives for visual diagrams in technical informatics education. Thus, we address existing gaps and foster inclusive learning environments that effectively support both visually impaired and sighted students.

III. Methodological Approach

This study follows an exploratory case study design aimed at developing and evaluating text-based alternatives for graphical representations in technical informatics education. The research was conducted over three semesters at [the University], where blind and visually impaired students were enrolled in core technical courses.

The methodological approach consisted of a continuous design and implementation cycle, including the development of accessible teaching materials (e.g., text-based Karnaugh-Veitch diagrams, UML class diagrams using PlantUML, Java-based logic gate models, and LTSpice NetLists). These materials were integrated into regular courses, and their usability was evaluated through iterative feedback.

Feedback was collected from two blind students, referred to as Student A and Student B, using structured interviews. Their insights helped assess the comprehensibility, practicality, and educational effectiveness of the proposed solutions. The student testimonials included in this paper were originally provided in German and were translated into English for inclusion.

The philosophical stance guiding this work can be described as pragmatic, focusing on solving concrete accessibility challenges in the educational context and prioritizing practical outcomes over theoretical generalizations.

IV. Proposed Solutions

The following subsections present our solutions for making diagrams and electronic circuits accessible to visually impaired people.

A. Accessibility to Diagrams for Visually Impaired Individuals

The primary objective was to develop and employ textual representations as an alternative to graphical representations for sighted students, ensuring accessibility and inclusivity.

We set the following requirements:

- Purely textual representations.
- Concise communication of the concepts conveyed by graphical representations.
- Usability of textual representations by both sighted and visually impaired students.
- Seamless content exchange between sighted and visually impaired students during lectures.
- Use of design and simulation tools that support textual model descriptions.

The proposed solutions were developed iteratively over the past two years and tested in lectures. Collaboration with blind students and the Study Center for Blind and Disabled Students (BliZ) ensured their relevance and effectiveness. The different approaches for each type of graphical representation are introduced in the following Subsections.

1) Karnaugh-Veitch Diagrams (KV Diagrams): KV diagrams simplify Boolean functions and minimize digital logic circuits. They were encoded as Markdown tables.

As an example, the truth table for a Logic NAND Gate which is a combination of a digital logic AND gate and a NOT gate connected together in series.

```
| Nr | B | A | Y |
|----|---|---|---|
| 0  | 0 | 0 | 1 |
| 1  | 0 | 1 | 1 |
| 2  | 1 | 0 | 1 |
| 3  | 1 | 1 | 0 |
```

A KV Diagram for the NAND:

```
|   Y  | !A  | A   |
|------|-----|-----|
| !B   | 1   | 1   |
| B    | 1   | 0   |
```

The Boolean expression of the NAND operation is:

$$Y = A \text{ NAND } B = \overline{A \wedge B}$$

In addition to the actual logical operation, the LaTeXcode of this formula includes additional rendering information. This affects the readability for blind people.

```
\[
Y = A \text{ NAND } B = \overline{A \land B}
\]
```

We used a simplified text version for Boolean expressions:
Y = A NAND B = !(A*B)
This simplified form is particularly barrier-free.

2) UML Class Diagrams: In the lectures *Programming* and *Technical Informatics*, UML class diagrams are introduced to visualize the structure of individual classes (name, attributes, methods, visibility) and relationships between classes.

To make these concepts independent of graphical representations, structural elements, such as attributes, methods, and class relationships (e.g., aggregation, composition, inheritance), were emphasized in textual form.

In a search, the following open source tools were found that enable barrier-free access to UML diagrams:

- *UML4ALL* (https://www.uml4all.net)
- *PlantUML* (https://plantuml.com)

Both tools were evaluated. *PlantUML* was chosen for its simplicity and browser-based functionality. *PlantUML* specifies a textual description of classes and provides a rendering tool which generates class diagrams from the textual description.

Here is an example of the textual description and the resulting diagram (Figure 1).

```
@startuml
class Person{
+ firstname. String
+ name: String
}
class Student {
+ matrNr: int
}
Person ^-- Student

class Lecturer {
+ departmentmatrNr: String
}
Person ^-- Lecturer

class Course {
+ title: String
+ lecturer: Lecturer
}
Course o-- Lecturer
@enduml
```

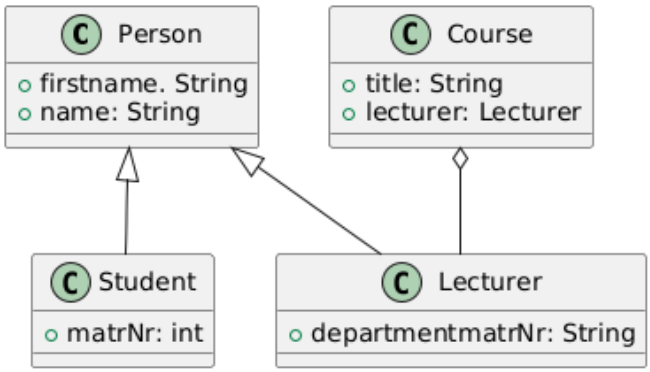

Figure 1. Generated class Diagram.

Class diagrams were presented in textual form in lecture materials, exercises, and exams, ensuring consistency between textual and graphical formats.

3) Logic Gates: Several tools are available for drawing and simulation of logic gates and circuits.

We use the Web-based tool *Circuit.js* for the development and simulation of circuits. Figure 2 shows a logic circuit of a full adder.

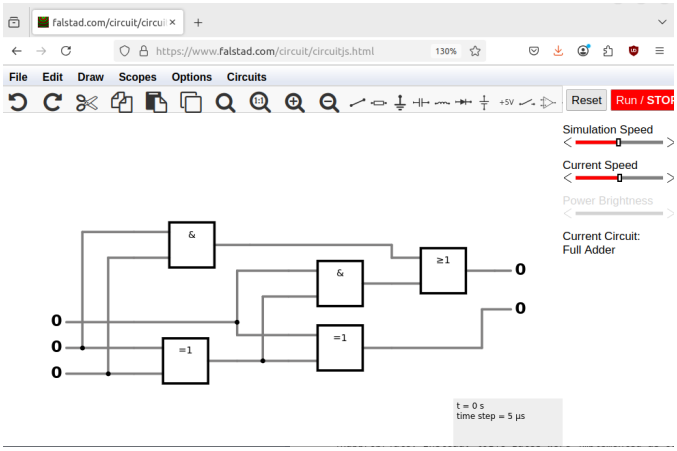

Figure 2. Circuit of a Full Adder.

Because this simulation tool works purely graphically, it is not barrier-free.

Textual specifications of logic gates and circuits are done by using Hardware Description Languages (HDL), like *Very High Speed Integrated Circuit Hardware Description Language* (VHDL).

However, introducing HDL in the first semester alongside Java was deemed pedagogically inappropriate. Instead, logic gates were implemented as Java classes.

As an example, this listing shows the code for an AND gate:

```
class AndGate {

    private boolean x1;
    private boolean x2;
    private boolean y;

    public void setX1(boolean x1)
        { this.x1 = x1; }

    public void setX2(boolean x2)
        { this.x2 = x2; }

    public boolean calculate() {
        y = x1 && x2;
        return y;
    }
}
```

Students built circuits as interacting objects and validated their behavior by printing out truth tables in the main() method.

B. Development and Simulation of Electronic Circuits for Visually Impaired Individuals

1) Motivation and Accessibility Challenges: Traditional graphical circuit schematics pose significant challenges for visually impaired students, as they rely heavily on visual elements that are difficult to interpret without specialized assistive technologies.

While previous sections have demonstrated how digital logic circuits can be implemented and simulated using Java, this approach does not address analog circuits. To fill this gap and enhance conceptual understanding for blind students, LTSpice is introduced as an accessible tool.

LTSpice, a widely used circuit simulation tool, presents additional obstacles due to its graphical interface, which is not optimized for screen readers or other accessibility tools [6]. To overcome these limitations, a text-based alternative is required, allowing visually impaired students to design and simulate electronic circuits effectively using structured NetLists instead of graphical schematics. An alternative option, Verilog-A (Verilog Analog), was considered; however, its complexity makes it unsuitable for first- and second-semester students.

This approach ensures inclusivity by enabling all students to participate in circuit analysis and development using accessible tools, while maintaining an appropriate level of complexity for introductory courses. Figure 3 shows a screenshot of a simple analog-circuit in LTSpice and its simulation-output in the waveform viewer.

2) NetList-Based Circuit Design: The NetList format provides a structured, text-based alternative to traditional graphical schematics, allowing visually impaired students to design and simulate electronic circuits efficiently. Unlike graphical editors, which require users to manually position and connect components, NetLists define circuits using a standardized

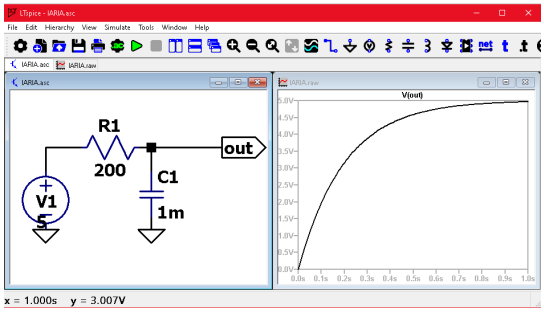

Figure 3. Example of an LTSpice Circuit Schematic.

syntax that can be easily interpreted by screen readers and assistive technologies.

A NetList consists of:

- **Component Definitions**: Each electrical component, such as resistors, capacitors, and voltage sources, is explicitly declared along with its respective parameters.
- **Node Connections**: Components are connected using numerical node identifiers, forming a network that describes the electrical relationships between circuit elements.
- **Simulation Directives**: Commands that instruct LTSpice on how to process the circuit, such as transient analysis settings.

Below is the circuit shown in Figure 3 in its simple NetList (.net) representation:

```
* voltage source (5) to Ua and ground (0)
V1 Ua 0 5
* Resistor (200 Ohms)
* from node Ua to node out
R1 Ua out 200
* Capacitor (1mF) to node out and ground (0)
C1 out 0 1m
* Simulate for 1 Second
.tran 1
```

This format ensures accessibility by allowing circuits to be designed, edited, and simulated using simple text files, which can be read and modified with any text editor. The structured approach also enables integration with screen readers, making circuit design more inclusive for visually impaired students.

3) Simulation in LTSpice using NetLists: LTSpice allows for the execution of circuit simulations using NetLists without relying on its graphical schematic editor, making it an accessible tool for visually impaired students. The simulation process can be carried out using command-line operations or script execution, ensuring a non-graphical workflow.

To import and simulate a NetList in LTSpice, follow these steps:

- Open the NetList file (.net) in a text editor and ensure its correctness.
- Load the NetList in LTSpice using the command-line interface.
- Execute the simulation using predefined analysis directives (e.g., `.tran` for transient analysis, `.ac` for frequency response).

Upon completion, LTSpice generates structured text-based simulation results. These outputs can be saved in CSV format or parsed into structured tables, allowing for further processing by screen readers or custom analysis tools for visually impaired users.

4) Alternative Output Formats for Accessibility: One of the major challenges for visually impaired students using LTSpice is the lack of accessibility in the waveform viewer. To address this, simulation results can be exported in text format, enabling alternative representations that are more accessible.

- Text Export via `.print` Commands:

```
.print tran V(Ua) V(out)
```

  The output is stored as a CSV file, allowing further processing with text editors or screen readers.
- Structured Table Representation: Instead of visual graphs, data can be displayed in a structured tabular format:

Time (s)	V(Ua)	V(out)
0.0	5.00	0.00
1.0	5.00	3.25
2.0	5.00	4.75

- Alternative Representations for Accessibility:
  - *Auditory Representation*: Using screen readers to read out the values sequentially.
  - *Tactile Representation*: Converting CSV data into a Braille format for students using Braille displays.
  - *Speech Synthesis*: Generating spoken descriptions of circuit behavior based on data trends.

5) Comparison: NetList vs. Graphical Schematic Input: The use of NetLists as an alternative to graphical circuit schematics presents both advantages and challenges.

Advantages:

- Provides **full accessibility** for visually impaired students, as it eliminates the reliance on visual representations.
- Allows **easy editing** using text editors with screen reader support.
- Enables **collaboration** with sighted students, as NetLists can be converted into graphical schematics when necessary.

Challenges:

- Requires learning a **specific syntax**, which might be an initial hurdle for some students.
- Does not support **intuitive spatial representation** of circuit layouts, making it harder for sighted individuals accustomed to schematic views.

6) Future Improvements: Several enhancements can be developed to further improve accessibility in electronic circuit design and simulation:

- **Integration of automatic speech synthesis**: This would allow simulation results to be read aloud, enabling better analysis for visually impaired students.
- **Development of haptic feedback solutions**: Tactile output devices could be used to provide a physical representation of circuit behavior.
- **Implementation of a user-friendly interface**: Creating an optimized LTSpice interface tailored to the needs of visually impaired users could enhance usability and efficiency.
- **Conversion tools for bidirectional translation**: Software that converts NetLists into graphical schematics and vice versa would support collaboration between sighted and visually impaired students.

By implementing these improvements, LTSpice can become a more inclusive tool that ensures equal opportunities for all students in electronic circuit design and simulation.

V. Feedback from Students

We asked two blind students for a statement comprising their experiences with the described approaches.

Student A:

> Experiences with barrier-free access to course material - Computer Engineering I in summer term 2024
>
> As a visually impaired student at the university, I attended the module **Computer Engineering I** in the summer term 2024, which was offered by [the professor].
>
> I would like to share my experiences with barrier-free access to the course content.
>
> As for the lecture slides during the semester, [the professor] kindly provided them to me in Markdown format. In the lectures, we used the open-source project Etherpad, which proved to be a very suitable solution for sharing notes in the classroom.
>
> Another approach to teaching logic gates during the semester was to use LTSpice-Netlists to simulate circuits. We then implemented the gates as C++ classes, which contributed significantly to the deepening and better understanding of the material.
> Finally, we worked with an implementation of the Von Neumann Machine Simulator, which provided a good insight into the processes of programs at machine level.
> All tools and alternative solutions used were tested with NVDA under Windows and worked perfectly. With regard to platform independence, it should be mentioned that the LTSpice application does not run directly under Linux. However, there are command line alternatives that can be used as a replacement.

Student B:

> Although visual concepts such as class diagrams were sometimes taught in Object-Oriented Programming (OOP), the lecturer managed to make them clear to me using PlantUML.
>
> The Etherpad helped me a lot to follow the blackboard notes or presentation and to understand class diagrams.

Original text in German, here translated. Student names replaced by 'A' and 'B'.

VI. Conclusion and Future Work

In this paper, we have introduced and evaluated a suite of text-based representations designed to make graphical diagrams in technical informatics education accessible for blind and visually impaired students. Our proposed solutions include PlantUML for a textual specification of UML class diagrams, simplified netlists for electronic circuit schematics, Java-based models for logic circuits, and structured Markdown representations for Karnaugh-Veitch diagrams.

Empirical evaluation through structured interviews demonstrated that these text-based approaches are comprehensible, practical, and pedagogically effective, significantly enhancing accessibility and inclusivity in mixed-ability classrooms. The significance of these findings is manifold. Primarily, our work contributes towards reducing educational barriers, thus aligning with broader inclusivity mandates such as those outlined in the UN-CRPD.

The demonstrated efficacy of text-based solutions establishes a practical foundation for scalable and sustainable educational practices, making complex technical content universally accessible without relying heavily on costly, specialized technology. However, our approach is not without limitations. Text-based representations require initial familiarization with specific syntactic conventions, which can pose challenges to some students. Additionally, the loss of intuitive spatial layouts inherent in graphical schematics might complicate conceptual understanding, particularly for sighted students accustomed to visual aids.

Future work should focus on addressing these limitations and further enhancing the usability and effectiveness of text-based approaches. Possible research directions include the development of hybrid solutions integrating text-based methods with haptic or auditory feedback, thus combining accessibility with intuitive comprehension. Additionally, automated tools facilitating bidirectional conversion between textual representations and graphical diagrams could foster better collaboration between sighted and visually impaired students. By addressing these challenges, we can continue advancing towards genuinely inclusive technical informatics education, enabling equitable learning opportunities for all students.

References

[1] S. A. Doore, J. K. Dimmel, T. Kaplan, B. A. Guenther, and N. A. Giudice, “Multimodality as universality: Designing inclusive accessibility to graphical information,” *Frontiers in Education*, vol. 8, 2023.

[2] F. Wildhaber, N. Salloum, M. Gygli, and A. Kennel, “Self-directed creation and editing of uml class diagrams on mobile devices for visually impaired people,” in *IEEE Tenth International Model-Driven Requirements Engineering (MoDRE)*, 2020.

[3] E. Pissaloux et al., “A new approach to assist virtual image accessibility for visually impaired people,” in *Universal Access in Human-Computer Interaction. User and Context Diversity*, M. Antona and C. Stephanidis, Eds. Cham: Springer International Publishing, 2022, pp. 251–261.

[4] J. Ducasse, M. J.-M. Macé, and C. Jouffrais, “From open geographical data to tangible maps: Improving the accessibility of maps for visually impaired people,” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2015.

[5] J. Ducasse, A. Brock, and C. Jouffrais, *Accessible Interactive Maps for Visually Impaired Users*. Springer International Publishing, 01 2018, pp. 537–584.

[6] Analog Devices Inc., “Ltspice - spice simulation software,” 2025, available online at: https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html, retrieved: May, 2025.