Accessible Representations of Visual Artifacts in Technical Informatics Education

University of Applied Sciences Mittelhessen, Germany

Diethelm Bienhaus, Michael Kreutzer, Florian von Zabiensky

18.05.2025

Presenter and Research Topics

- Prof. Dr.-Ing. Dipl.-Wirt. Ing. Diethelm Bienhaus
  - Professor for Computer Science / Embedded Systems
  - University of applied sciences / Technische Hochschule Mittelhessen
- Research topics of the Institute of Technology and Informatics (ITI)
  - Assisting Systems / Autonomous Vehicles / Edge AI / Remote Labs / ...

The presentation is generated from Markdown text. This is available under:

Introduction

- Visual diagrams (e.g., UML, KV diagrams, circuit schematics) are essential in engineering education.
- Critical accessibility issues for blind and visually impaired learners.
- Compliance with UN-CRPD requires inclusive education.
- Main research questions:
  - How to translate graphical formats into **equivalent text-based forms**?
  - Can these alternatives be both **inclusive** and **pedagogically sound**?
- Our appoach: **text-based alternatives** to graphical artifacts used in Technical Informatics.
- Developed and tested over 3 semesters with blind students at THM.

Alternative Approaches / Related Work

- Multimodal & Universal Design
  - Use of vibro-audio interfaces.
  - Effective but **hardware-intensive** and **costly**.
- Text-Based Tools
  - Wildhaber et al.: Accessible mobile UML editors.
  - Tools like **PlantUML** reduce visual dependency.
- Tactile Diagrams
  - Thermo-formed paper.
  - Good for haptics, but **inflexible and expensive**.
- Tangible Interfaces
  - Interactive physical maps.
  - High potential, but need **space and tech**.

Methodological Approach

- **Exploratory case study** across 3 semesters.
- Iterative development of text-based content (UML, KV, Logic Gates, Circuits).
- Evaluation through interviews with 2 blind students (A & B).
- Focused on **practical accessibility** rather than theory.

Proposed Solutions

Requirements

- Text-only, concise, and semantically equivalent.
- Usable by sighted and blind students.
- Compatible with teaching tools and platforms.

Karnaugh-Veitch (KV) Diagrams

NAND Truth Table

Nr	B	A	Y
0	0	0	1
1	0	1	1
2	1	0	1
3	1	1	0

KV Table Representation

Y	!A	A
!B	1	1
B	1	0

Boolean Expression: Y = A NAND B = !(A * B)

UML Class Diagrams

- Tools evaluated:
  - UML4ALL
  - **PlantUML** (chosen for simplicity)

Example PlantUML Code

```
@startuml
class User {
  - id: int
  - name: string
  + login()
}
@enduml
```

Example of a plantUML Class Diagram Description I

```
@startuml
class Person{
+ firstname. String
+ name: String
}

class Student {
+ matrNr: int
}
Person ^--  Student

class Lecturer {
+ departmentmatrNr: String
}
```

Example of a plantUML Class Diagram Description II

```
Person ^--  Lecturer

class Course {
+ title: String
+ lecturer: Lecturer
}

Course o-- Lecturer
@enduml
```

Generated Class Diagram

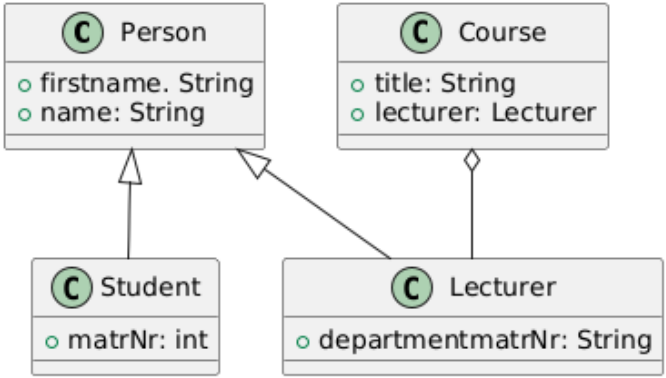

Figure 2: UML Class Diagram

Text description and graphic used in lecture slides, exercises, and exams.

Logic Gates

- Problem: GUI-based tools like Circuit.js not accessible.
- Solution: **Java Classes for Logic Gates**

Java Code (Simplified)

```
class AND {
  boolean compute(boolean a, boolean b) {
    return a && b;
  }
}
```

- Students validated logic with **truth tables** in `main()`.

LTSpice Challenges

- Not screen-reader-friendly.
- GUI-based waveform viewer not accessible.

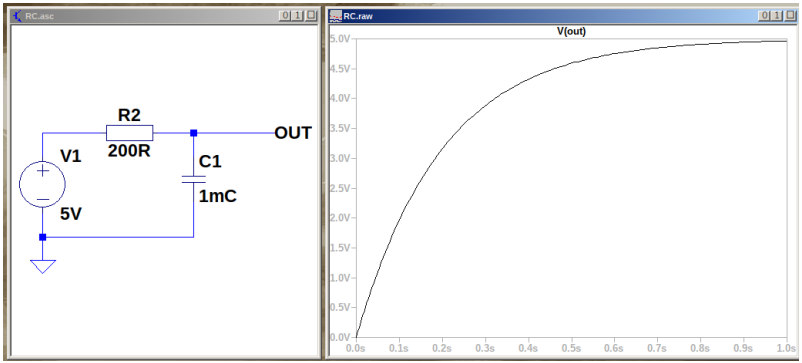

Figure 3: LTSpice

Description of Figure 2 (I/II)

A textual description of the figure generated by chatGPT:

Left Side (Circuit Diagram)

- **Software**: LTspice (a circuit simulation tool)
- **Circuit Components**:
  - **Voltage Source (V1)**: Supplies 5 volts (DC).
  - **Resistor (R1)**: 200 ohms.
  - **Capacitor (C1)**: 1 millifarad (1mF).
  - The resistor and capacitor form a simple **RC charging circuit**.
  - The capacitor is connected between the resistor and ground.
  - The output voltage is measured at the node between the resistor and capacitor, labeled **"out"**.

Description of Figure 2 (II/II)

Right Side (Simulation Plot)

- The graph shows the **voltage across the capacitor (V(out))** as it charges over time.
- **X-axis**: Time (from 0 to 1 second)
- **Y-axis**: Voltage (from 0V to 5V)
- The curve represents the typical **exponential charging behavior** of a capacitor, approaching 5V.
- At $\mathbf{t = 1}$ **second**, the output voltage is approximately **3.007V** (as shown at the bottom of the plot window).

This is a classic RC charging example used to demonstrate time-domain response in first-order circuits.

Caveat: Generated Descriptions

chatGPT:

- At $\mathbf{t = 1\ second}$, the output voltage is approximately $\mathbf{3.007V}$ (as shown at the bottom of the plot window).

Really?

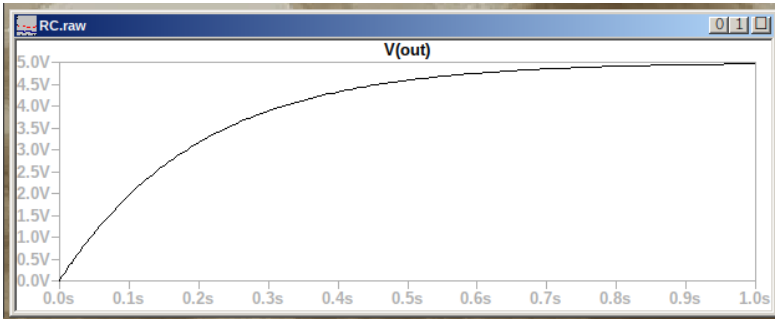

Figure 4: LTSpice RC Time Graph

Description of the Right Side of the Image

The right side of the image shows the **voltage response (V_{out})** of an RC (resistor-capacitor) circuit over time.

Details:

- **Graph Title:** `V(out)`
- **X-axis:** Time (in seconds), range: 0 to 1.0 s
- **Y-axis:** Voltage (in volts), range: 0 V to 5 V
- **Curve Shape:** The curve shows an **exponential rise** in voltage, starting at 0 V and asymptotically approaching 5 V.

Interpretation:

This is the typical behavior of a **charging capacitor** in an RC circuit:

Improved Answer II

- At time (t = 0), the capacitor is uncharged, so ($V_{\text{out}} = 0$).
- Over time, the voltage across the capacitor increases as it charges through the resistor.
- The voltage follows the formula:

$$V(t) = V_{\text{in}} \cdot \left(1 - e^{-t/RC}\right)$$

where: - ($V_{\text{in}} = 5$, V) - (R = 200 , Ω) - (C = 1 , mF = 0.001 , F) - (RC = 0.2 , s) is the time constant

After approximately (t = 1 , s) (five times the time constant), the voltage closely approaches the final value of 5 V, which is expected behavior for this type of response.

Text-Based Alternative: NetList

- NetList Features
  - Text format for:
    - Components
    - Node connections
    - Simulation directives

Example: NetList of Circuit shown in Figure 2

```
V1 N001 0 5V
R2 OUT N001 200R
C1 OUT 0 1mC
.tran 1 startup
.end
```

Simulation Output

Text Export

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

CSV Output (Example)

```
Time (s), V(in), V(out)
0.0,      5.00,  0.00
0.5,      5.00,  4.52
1.0,      5.00,  4.98
```

Accessible Alternatives

- Screen readers
- Braille displays
- Speech synthesis

Student Feedback I

Student A: 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 Feedback II

Student B: Although visual concepts such as class diagrams were sometimes taught in 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.

Conclusion

- Developed inclusive, **textual alternatives** for:
  - UML (via PlantUML)
  - KV diagrams (Markdown)
  - Logic circuits (Java)
  - Electronic schematics (NetList for LTSpice)
- Ensured **semantic equivalence** & **educational effectiveness**.

Future Work

- Speech synthesis of simulations
- Haptic feedback for circuit behavior
- Simplified accessible LTSpice interface
- Tools for converting between text and graphical formats
- Usage of generative artificial intelligence to generate
  - Podcasts from teaching materials and textbooks
  - validation of textual descriptions of graphics
- Use of automata description languages to generate state charts
  - blind students work with the textual specification
  - sighted peers view the rendered automaton