ISSN: 1692-7257 - Volume 2 – Number 46 - 2025

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v2i46.4126

Voice Driven UML Modeling System for Visually Impaired Students in Software Engineering Education

Sistema de modelado UML controlado por voz para estudiantes con discapacidad visual en la formación en Ingeniería de Software

PhD. Carlos Henriquez Miranda[1], Ing. Malak Andres Sanchez Cataño[1], PhD. German Sanchez-Torres[1]

[1] ***Universidad del Magdalena,*** *Facultad de Ingeniería, Grupo de Investigación y Desarrollo en Sistemas y Computación, Santa Marta, Magdalena, Colombia.*

Correspondence: *chenriquezm@unimagdalena.edu.co*

Received: *march 25, 2025.* ***Accepted:*** *july 29, 2025.* ***Published:*** *august 06, 2025.*

How to cite: *C. Henriquez Miranda, M. A. Sanchez Cataño, and G. Sanchez-Torres, "Voice Driven UML Modeling System for Visually Impaired Students in Software Engineering Education", RCTA, vol. 2, no. 46, pp. 171–180, Aug. 2025. Recovered from https://ojs.unipamplona.edu.co/index.php/rcta/article/view/4126*

Abstract: Visual impairment limits students' access to object-oriented modeling environments that rely on graphical interfaces. Advances in Text-to-Speech (TTS) and Speech-to-Text (STT) technologies make it possible to consider mechanisms to compensate for this barrier, yet their adoption in educational platforms lacks evidence-based guidelines. This work aims to design, implement, and validate a voice-controlled UML modeling system that enables visually impaired students to create, edit, and query class and use-case diagrams. A four-phase methodology was followed: (i) requirements elicitation; (ii) modular architectural design with grammatical validation and the use of large language models (LLMs); (iii) implementation in Python 3.11, FastAPI, and cloud-based STT/TTS services; and (iv) technical evaluation using recognition accuracy, latency, and task-time metrics. The prototype executed 20 critical commands with a recognition accuracy of 97 % and maintained full syntactic coherence in the UML models generated. The average times to complete creation, editing, and navigation tasks were 4.23 s, 6.78 s, and 5.24 s, respectively. The average TTS latency (1,468 ms) exceeded the 500 ms target, identifying the NLP module as the main bottleneck. The system demonstrates technical feasibility and adheres to the defined accessibility guidelines (WCAG 2.2). Future improvements will focus on reducing TTS latency, expanding the command repertoire, and conducting large-scale usability evaluations (SUS).

Keywords: accessibility, speech synthesis, UML modeling.

Resumen: La discapacidad visual limita el acceso de los estudiantes a entornos de modelado orientado a objetos que dependen de interfaces gráficas. El avance en las tecnologías Text-to-Speech (TTS) y Speech-to-Text (STT) permite pensar en mecanismos para compensar esta barrera, pero su adopción en plataformas educativas carece de directrices basadas en evidencia. Este trabajo se orienta hacia diseñar, implementar y validar un sistema de modelado UML controlado por voz que permita a estudiantes con

discapacidad visual crear, editar y consultar diagramas de clases y de casos de uso. Se siguió una metodología de cuatro fases: (i) levantamiento de requisitos; (ii) diseño arquitectónico modular con validación gramatical y uso de LLMs; (iii) implementación en Python 3.11, FastAPI y servicios de STT/TTS en la nube; (iv) evaluación técnica empleando métricas de precisión de reconocimiento, latencia y tiempos de tarea. El prototipo ejecutó 20 comandos críticos con una precisión de reconocimiento del 97 % y mantuvo coherencia sintáctica total en los modelos UML generados. El tiempo medio para completar tareas de creación, edición y navegación fue de 4,23 s, 6,78 s y 5,24 s, respectivamente. La latencia promedio de TTS (1 468 ms) superó el objetivo de 500 ms, identificando al módulo PLN como principal cuello de botella. El sistema demuestra viabilidad técnica y sigue lineamientos de accesibilidad definidos (WCAG 2.2). Las mejoras futuras se centrarán en reducir la latencia TTS, ampliar el repertorio de comandos y realizar evaluaciones de usabilidad (SUS) a gran escala.

Palabras clave: accesibilidad, síntesis de voz, modelado UML.

1. INTRODUCTION

Visual impairment represents a significant barrier to educational inclusion and independent learning today. According to the World Health Organization, more than one billion people have some degree of visual deficiency [1], which restricts their access to printed texts, graphical interfaces, and conventional educational environments.

In this context, assistive auditory technologies, particularly text-to-speech (TTS) synthesis, emerge as a key resource for converting textual content into audible information, enhancing cognitive accessibility and the autonomy of students with visual impairments. The growing adoption of online education platforms and remote learning environments increases the need for interactive vocal feedback mechanisms—such as reading texts, quizzes, and exams—to ensure equal learning opportunities.

Although voice feedback is well established in navigation applications for independent mobility, its integration into educational platforms requires a thorough analysis of interaction dynamics, multisensory synergy, and the measurement of pedagogical outcomes. Additionally, regulatory frameworks such as the Web Content Accessibility Guidelines (WCAG) [2], demand not only the incorporation of TTS functions but also their coherence in pedagogical workflows and interactive interfaces; however, they lack evidence-based guidelines on their effective application.

Furthermore, recent research on interactive technologies for users with visual disabilities shows valuable technical and interaction advancements, but it falls short in establishing a comprehensive approach that articulates designs, architectures, and evaluation metrics.

In some specific contexts, significant developments in this direction include: in mobility, wearable devices combine auditory feedback with ultrasonic and infrared sensors, yielding good results but with weaknesses in studies on their safety in real-world environments [1].

In the field of remote music education, Networked Music Performance platforms prioritize low latency and high sound fidelity for students with multiple disabilities, but omit TTS schemes that facilitate real-time textual feedback [3]. Similarly, in the learning of virtual textures, vibrotactile stimuli and voice synthesis have been integrated for material classification, showing potential for non-visual recognition but limiting themselves to non-textual domains [4].

Despite the aforementioned contributions, weaknesses still persist. There are no comprehensive comparisons of usability criteria nor evaluations of effectiveness in learning performance. Without a comparative framework, developers and educators lack objective criteria to select and adjust TTS technologies that optimize both user experience and academic outcomes [5].

In this context, the present work proposes a voice-activated UML modeling system, specifically designed for students with visual impairments in software engineering education environments. Unlike general approaches based solely on content reading, this proposal focuses on bidirectional interaction through verbal commands that allow the

creation, editing, and consultation of UML diagram elements. The initiative responds to the need to integrate TTS and STT technologies in learning environments, where vocal feedback not only converts text to audio but actively participates in navigation and the construction of technical knowledge. To this end, a methodology was developed, structured in four phases, ranging from requirement gathering to technical validation.

This document is structured as follows: Section 2 describes work in three different areas. Section 3 outlines the methodology used for prototype development. Sections 4, 5, and 6 present the results, discussion, and conclusions, respectively.

2. TECHNOLOGICAL BACKGROUND

Recent developments in assistive technologies for people with visual impairments are grouped into three complementary areas.

2.1 Non-Educational Assistive Technologies

Portable orientation and mobility devices for people with low vision combine readings from RGB-D/stereo cameras and ultrasound, providing auditory or haptic feedback with high precision, but are weak in standardized safety metrics and validations in dynamic environments [1].

Other works explore vibrotactile displays where a Convolutional Neural Network (CNN) smooths contours and translates them into directional patterns that accelerate exploration [6], and 16 × 16 MEMS Braille displays coupled with haptic-auditory environments, which, using Seq2Seq + WaveNet models, recognize characters with 97% accuracy in 0.5 seconds [7].

Portable robotic guides trained using Human Path Prediction Network (HPPN) for trajectory estimation, employing a Covariance Matrix Adaptation Evolution Strategy (CMA-ES), achieve reliable trajectories with 1,507 real episodes [8].

In space-touch interaction, MapIO [9] combines physical maps, voice synthesis, and LLM-based dialogue, improving System Usability Scale (SUS) response accuracy at the cost of higher inference latency. The Force-Feedback Tablet (20 × 20 cm, <1 kg) uses a flat thumbstick to generate haptic effects (friction, edges, attraction) and reduces guided exploration time by 44%, with tests limited to the laboratory [10]. Finally, the VIS4ION system integrates 5G sub-6 GHz/mmWave edge computing to process high-resolution vision with global latency <100 ms, though energy consumption and ergonomics in real-world scenarios still need evaluation [11].

2.2 Educational Tactile and Haptic Interfaces

Tactile graphics (TG) have been proposed and used, which translate images into relief through dynamic screens, vibrotactile actuators, and force feedback to represent mathematical structures and curves [12].

In music, the challenges focus on latency. Low-latency platforms like LOLA, JackTrip, SoundJack, JamKazam, SonoBus, or FarPla aim for very low latencies, similar to in-person ones ($L \leq 30$ ms), but require dedicated hardware and still lack fully accessible interfaces [3]. Similarly, a multimodal haptic engine combining Informer and VGG16 generates high-fidelity signals in 45–57 ms with 93,33% accuracy by combining vibration and TTS [4].

In education and rehabilitation, the Hyperbraille display [13] (30 × 32 taxels, 5 Hz) reports improvements of 33–68% in shape recognition and 41–61% in spatial memory. Tools like TAURIS [12] allow exploring pre-labeled graphs with contextual narration, surpassing traditional Braille and screen readers. A recent taxonomy maps solutions for access to and creation of mathematical content, highlighting the prevalence of TTS and standards such as MathML and EPUB3 Similarly, GraficiAccessibili [14] employs rhythmic sonification and vibration to represent functions on Android tablets, achieving correct identification after brief training, though limited to simple functions. Tangible interfaces, e.g., Tac Trace [15], use 3D tokens tracked by Trackmate vision to provide Arabic audio and teacher tracking.

Finally, the application of asymmetric vibrations in white canes significantly reduces the Root Mean Square Error (RMSE) of sweep width during the touch technique learning process, enabling remote training without physical contact with the specialist [16]. Finally, the application of asymmetric vibrations in white canes significantly reduces the RMSE of sweep width during the touch technique learning process, enabling remote training without physical contact with the specialist.

2.3 Multimodal Curriculum and Assessment Tools

This line of research investigates instructional environments that synchronize audio, touch, and visualization with automated feedback, structured dialogue, and learning analytics. Their frameworks are typically based on Design-Based Research cycles and support TTS/STT and screen readers to maximize accessibility and personalization.
In higher education, digital peer assessment (DPA) enables students to evaluate their peers' work while developing critical thinking skills [17].

Accessible Design-Based Research frameworks integrate audio and haptics. For example, Eyeland achieved a 91.7% satisfaction rate and improved performance for both users with visual impairments and sighted individuals [18].

A multilingual accessible writing system, based on Random Forest (f1 = 0.998) and a confidence threshold > 70%, demonstrated the feasibility of incorporating automatic and auditory assessment in APD [19]. A conversational graph exploration system, based on rule-guided dialogue to describe finite automata, outperformed the standard HTML representation in an A/B test ($p < 0.05$), while generic LLMs failed to maintain consistency [20].

3. METHODOLOGY

The research was structured in four sequential phases, each aimed at a specific objective (see Fig 1):

- In Phase 1, requirement gathering and the definition of use scenarios for students with visual impairments in Unified Modeling Language (UML) modeling activities were carried out. Based on interviews with instructors and the analysis of curricula, the priority UML artifacts and the workflows that the prototype should support were identified.
- Phase 2 was dedicated to the architectural design of the system. A modular solution was defined, capable of receiving voice or text commands, processing their meaning through grammar rules and Natural Lenguage Processing (NLP) services, and managing the generation and persistence of diagrams in JSON format. Communication protocols, integration patterns, and accessibility criteria were established based on WCAG 2.2.
- During Phase 3, the prototype was implemented and integrated using Python 3.11. FastAPI was employed for the web service, SQLite as a lightweight storage solution, and independent adapters for the STT, TTS, and LLM services. Continuous delivery and automated testing ensured code quality and the stability of the REST and WebSocket interfaces.
- In Phase 4, the system evaluation with real users was programmed. A pilot study with students with visual impairments was designed to measure performance, command recognition accuracy, and usability using the SUS scale. The data obtained in this phase will guide the fine-tuning of the components and NLU prompts.

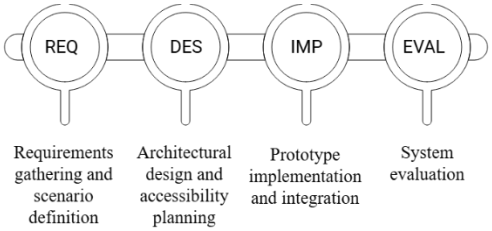

***Fig. 1**. Outline of the general methodology applied.*
***Source:** Own elaboration.*

4. RESULTS AND DISCUSSION

4.1 Requirement Gathering and Scenario Definition

During this phase, three semi-structured interviews were conducted with Software Engineering faculty members, complemented by the analysis of current curricula. From this analysis, it was concluded that, in undergraduate UML modeling courses, class diagrams and use case diagrams dominate the majority of practical activities. This prevalence is documented both in *curriculum mapping* studies for Software Engineering education [21], [22] and in the UML 2.5 specification itself [23], which recommends these artifacts to represent, respectively, the system's static structure and functional requirements.

Two main usage scenarios were defined for these two types of diagrams: on one hand, the creation of a model from scratch during a practical session, and on the other hand, the incremental editing of an already existing diagram. These scenarios allowed for the definition of an initial set of 41 voice commands organized into nine categories —seven global (e.g., start, save, or undo), six for navigation

and querying, and 28 specifics for CRUD operations (*Create, Read, Update, and Delete*) on classes and use cases— (see Table 1). The repertoire includes instructions such as "create a customer class," "add a name attribute to Customer," or "link the User actor to the Register use case".

To validate the minimum viable product (MVP), 20 of these commands (50% of the total) were implemented, focusing exclusively on operations related to class and use case diagrams. This approach ensures that the voice interface covers the critical tasks identified in the requirements gathering, in line with the recommendations of [24] regarding the prioritization of artifacts in educational environments.

Based on response time and accuracy metrics reported in the literature, the objective was set for each command to be processed in less than 2.0 seconds with a recognition accuracy rate of at least 90% in voice recognition [25]. Similarly, it was defined that the system must maintain a consistent internal model, apply UML syntactic rules, and expose the diagram structure in JSON format for persistence and export. In general, the established specification (see Table 2):

- Functional requirements:
    - Multimodal input (voice/text)
    - Processing time $\leq 2{,}0$ s
    - Accuracy ≥ 90 %
- Non-functional requirements:
    - End-to-end latency < 4 s on academic networks (< 10 Mbps)
    - Compliance with WCAG 2.2 (*Web Content Accessibility Guidelines*)
    - Accessible Rich Internet Applications (ARIA) navigation and labeling, ensuring traceability and compatibility with existing CASE tools.

Table 1: *Examples of CRUD commands from the prototype*

Commands	Category	Diagram
Create a class <Name>	CRUD class	Classes
Delete a class <Name>	CRUD class	Classes
Rename class <Old> to <New>	CRUD class	Classes
Add an attribute <Attribute> to <Class>	CRUD class	Classes
Add a method <Method> to <Class>	CRUD class	Classes
Undo the last action	Control	Both
Create a use case <Name>	CRUD use case	Use cases
Link actor <Actor> with use case <Name>	CRUD use case	Use cases
Save the diagram	Global	Both
Close the diagram	Global	Both

Source: *Own elaboration*

Table 2: *Defined requirements*

ID	Requirement	Type	Target Value
RF1	Interpret each command in less than 1.5 s	Functional	$< 1{,}5$ s
RF2	Minimum accuracy in voice recognition	Functional	≥ 92 %
RF3	Consistent internal model with UML syntactic rules	Functional	Full compliance
RF4	Expose diagram in JSON format for persistence/export	Functional	Valid JSON
RNF 1	End-to-end latency on academic networks (≤ 10 Mbps)	Non-functional	< 4 s
RNF 2	Compliance with WCAG 2.2 (contrast, ARIA, keyboard navigation)	Non-functional	Compliance with all criteria
RNF 3	Auditory feedback with explicit confirmations (TTS)	Non-functional	Confirmation after each action

Source: *Own elaboration*

4.2 Architectural design

The system was structured with a three-layer modular pattern — input, semantic processing, and generation — complemented by a presentation module (see Figure 2). Each layer groups components with well-defined responsibilities, facilitating traceability between requirements and design [26].

In the input layer, a Voice Command Interpreter capable of receiving both audio signals (via the STT service) and written text. In line with the Web Speech API for modern browsers, this component converts dictation into raw text and queues the requests for subsequent analysis. To ensure accessibility, all control elements are labeled according to WCAG and ARIA specifications, and it includes keyboard navigation.

The semantic processing layer integrates two subcomponents. First, a syntax validator based on *UML Grammar Rules* (implemented in JSON Schema) discards malformed structures and, if necessary, invokes a disambiguation engine that generates clarification requests to the user. Second, an external Natural Language Processing (NLP) service — implemented through calls to a *function calling* translates the validated text into CRUD operations on UML elements. This hybrid approach, combining controlled grammars and language models, balances precision and flexibility [27].

In the generation and persistence layer resides the *Diagram Orchestrator*, responsible for coordinating changes to the internal model and exposing it in JSON format through a REST API. For real-time notifications (e.g., view updates), a WebSocket channel is enabled. Persistence uses SQLite as a lightweight storage solution, facilitating export and integration with CASE tools.

Finally, the presentation module runs on a React client that consumes the REST API and the WebSocket channel to render SVG diagrams using PlantUML. This client plays TTS outputs to confirm each action and updates the interface without reloading, meeting the requirements for operability and robustness [28].

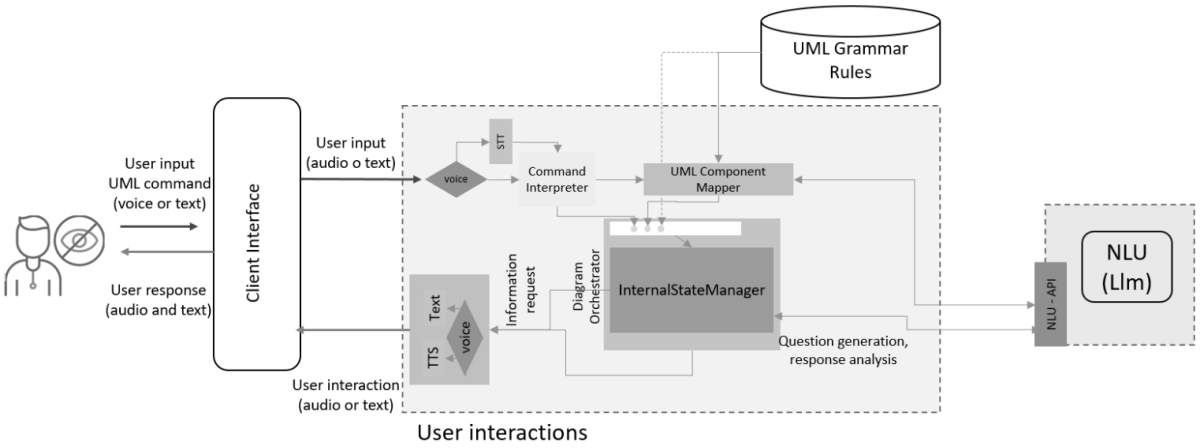

***Fig. 2**. Block diagram with three zones: (i) Accessible Input —the user dictates or writes a UML command; (ii) Semantic Processing —the command passes through the interpreter, UML mapping and a query to validate LLM with UML rules; (iii) Model Generation/Management —internal state persisted, and audio/text responses returned to the user.*
***Source:** Own elaboration.*

4.3 Implementation and Integration

The prototype was developed entirely in Python 3.11 and was organized into modules that cover everything from audio capture to the generation of the final diagram. The application starts in the main script, where credentials and configuration parameters are read, and instances of the following are created:

- *LLMInterface*, responsible for communication with the language model and managing the registration and invocation of tools through *function calling*.
- *InternalStateManager*, which maintains the internal state of the diagrams, logs each operation as a tool, and executes CRUD and export functions.
- *AudioHandler*, which delegates audio-to-text conversion and microphone audio capture to the STT_module and TTS_module.

In the main loop, the system alternates between text and voice modes; in the case of voice, *AudioHandler* listens to the PCM signal and converts it to WAV using Pydub before invoking the Google recognition service. The resulting text is sent to LLMInterface, which generates a structured response according to the JSON schemas defined in function_schemas.py. This response invokes the corresponding tool in InternalStateManager, which updates the internal model stored in SQLite and returns a confirmation message. Then, AudioHandler synthesizes the response into audio and plays it back to the user.

Persistence and interoperability with CASE tools are achieved by exporting the internal model to JSON, which transforms into a PlantUML script and processes via the command line to produce a diagram in PNG, SVG, or PDF format. Communication between back-end components is handled using FastAPI (REST endpoints for discrete operations) and a WebSocket server (for real-time

notifications), supported by HTTPX and AnyIO for asynchronous operation (see Table 3).

To facilitate extensibility and maintainability, each LLM tool is associated with a *JSON schema* that validates input parameters before executing the function. This pattern ensures that only requests with the expected structure are executed, reducing deserialization errors and enhancing the robustness of the prototype (Newman, 2015).

Table 3: *Implementation of Components and Technologies Used*

Module	Main Function	Tecnología
Audio Capture	Listens to keyboard events and records raw audio	pyaudio, keyboard
Speech Recognition (STT)	Converts audio to text using API	speech_recognition, Pydub
Text-to-Speech (TTS)	Generates audio from text using Google Cloud TTS or gTTS	google-cloud-texttospeech, gTTS, Pydub
Command Interpretation	Registers and validates function calling requests	OpenAI API, JSON Schema
Internal State Management	CRUD and diagram export, action orchestration	SQLite, Python dict
Diagram Export	Converts JSON to PlantUML and execute PlantUML via command line	PlantUML, subprocess
API REST and WebSocket	Exposes synchronous and asynchronous operations	FastAPI, HTTPX, AnyIO
User Interface (Client)	Dynamic SVG rendering and TTS playback	React, PlantUML

Source: *Own elaboration*

4.4 Validation

The evaluation focused solely on technical aspects by performing core tasks: creating a diagram from scratch, incremental editing, and navigating an existing diagram. To quantify the system's effectiveness and usability, the following metrics were defined:

- Time on task: the duration elapsed from the start to the completion of each task (in seconds). This metric evaluates workflow efficiency and is calculated as:

$$T_{task} = t_{fin} - t_{ini} \quad (1)$$

where t_{ini} and t_{fin} correspond to the timestamps marking the start and end of the task, respectively [28].

- Recognition Accuracy: the proportion of commands correctly interpreted by the system, defined as

$$A_{rec} = \frac{N_{Correctly}}{N_{total}} \times 100\% \quad (2)$$

where $N_{correctly}$ is the number of commands executed without recognition error, and N_{total} is the total number of commands issued by the user.

- Error Rate: the complement of accuracy, calculated as:

$$E = 100\% - A_{rec} \quad (3)$$

- TTS Latency: the elapsed time between the invocation of the synthesis function and the start of audio playback, measured in milliseconds.

$$L_{TTS} = t_{iniRep} - t_{iniInv} \quad (4)$$

where, $t_{iniRep}$is the playback start time and t_{iniInv} is the invocation start time.

See Table 4 for system metrics evaluation.

Table 4: *System Metrics Evaluation Table.*

Metric	m	ds	Obj
Creation Time (s)	4.23s	±1.8	–
Editing Time (s)	6.78s	±1.4	–
Navigation Time (s)	5.24s	–	–
Recognition Acc (%)	97 %	–	≥ 90 %
Error Rate (%)	3 %	–	≤ 10 %
TTS Latency (ms)	1468ms	±224	< 500 ms

ds: standard deviation

Source: *Own elaboration*

4.5 Discussion

The analysis of the results allows for the identification of relevant technical aspects and their implications within the domain of users with visual impairments:

The recognition accuracy reached 97%, exceeding the minimum threshold of 90% established in RF2. This level of accuracy indicates that the STT engine and the NLP engine, combined, provide adequate

reliability for most interactions, reducing the need for clarification requests to the user.

The average TTS latency of 1468 ms exceeds the target of 500 ms (RNF3), which prevents guaranteeing prompt auditory confirmations. The largest latency cost component is the NLP module.

The average times for the tasks of creation (4.23 s), editing (6.78 s), and navigation (5.24 s) reflect the entire interaction flow—from dictation to view update. Although these values exceed the command processing metric (< 1.5 s) defined in RF1, it is important to consider that they include NLP operations, persistence, and rule validation. For users with visual impairments, minimizing the number of steps in the workflow and optimizing the WebSocket channel can help reduce the total task time.

Full compliance with UML syntactic rules (RF3) and the generation of valid JSON (RF4) confirm the robustness of the semantic layer and the diagram orchestrator.

Accessibility**:** the adoption of WCAG 2.2 and ARIA for navigation and labeling ensures that the system meets the expected accessibility criteria. Although usability metrics were not quantified in this phase, technical compliance lays the groundwork for a future evaluation using the SUS scale.

5. CONCLUSIONS

The implementation of a coherent internal model and the exposure of diagrams in JSON format ensure interoperability with CASE tools and UML syntactic consistency. Although overall task times (4–7 s) exceed isolated command processing thresholds, the system's modularity enables optimization approaches focused on the orchestrator and the reduction of client-side operations.

In terms of application in educational settings, the system demonstrates technical feasibility to support UML modeling activities via voice, which could enhance autonomy and efficiency for students with visual impairments.

As future work, we propose extending the repertoire of CRUD commands as well as the types of diagrams supported. Additionally, optimizing the WebSocket channel and the interaction with the NLP module—where the greatest latency cost is concentrated—will be prioritized and conduct a usability evaluation using the SUS scale and adjust the NLU prompts based on feedback from real users.

REFERENCES

[1] A. D. P. D. Santos, A. H. G. Suzuki, F. O. Medola, y A. Vaezipour, «A systematic review of wearable devices for orientation and mobility of adults with visual impairment and blindness», *IEEE Access*, vol. 9, pp. 162306-162324, 2021, doi: 10.1109/ACCESS.2021.3132887.

[2] World Wide Web Consortium, «Web Content Accessibility Guidelines (WCAG) 2.2». diciembre de 2024.

[3] C. Rottondi, M. Sacchetto, L. Severi, y A. Bianco, «Toward an inclusive framework for remote musical education and practices», *IEEE Access*, vol. 12, pp. 173836-173849, 2024, doi: 10.1109/ACCESS.2024.3501414.

[4] D. Chen *et al.*, «Visually impaired people learning virtual textures through multimodal feedback combining vibrotactile and voice», *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 33, pp. 453-465, 2025, doi: 10.1109/TNSRE.2025.3528048.

[5] S. Raffoul y L. Jaber, «Text-to-Speech Software and Reading Comprehension: The Impact for Students with Learning Disabilities», *Canadian Journal of Learning and Technology*, vol. 49, n.º 2, Art. n.º 2, nov. 2023, doi: 10.21432/cjlt28296.

[6] D. Chen, J. Liu, L. Tian, X. Hu, y A. Song, «Research on the method of displaying the contour features of image to the visually impaired on the touch screen», *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 2260-2270, 2021, doi: 10.1109/TNSRE.2021.3123394.

[7] D. Chen *et al.*, «Development and evaluation of refreshable braille display and active touch-reading system for digital reading of the visually impaired», *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 32, pp. 934-945, 2024, doi: 10.1109/TNSRE.2024.3363495.

[8] H. -S. Moon y J. Seo, «Sample-efficient training of robotic guide using human path prediction network», *IEEE Access*, vol. 10, pp. 104996-105007, 2022, doi: 10.1109/ACCESS.2022.3210932.

[9] M. Manzoni, S. Mascetti, D. Ahmetovic, R. Crabb, y J. M. Coughlan, «MapIO: a gestural and conversational interface for tactile maps», *IEEE Access*, vol. 13, pp. 84038-84056, 2025, doi: 10.1109/ACCESS.2025.3566286.

[10] S. L. Gay, E. Pissaloux, K. Romeo, y N. - T. Truong, «F2T: a novel force-feedback haptic architecture delivering 2D data to visually impaired people», *IEEE Access*, vol. 9, pp. 94901-94911, 2021, doi: 10.1109/ACCESS.2021.3091441.

[11] Z. Yuan *et al.*, «Network-aware 5G edge computing for object detection: Augmenting wearables to "see" more, farther and faster», *IEEE Access*, vol. 10, pp. 29612-29632, 2022, doi: 10.1109/ACCESS.2022.3157876.

[12] M. Zeinullin y M. Hersh, «Tactile audio responsive intelligent system», *IEEE Access*, vol. 10, pp. 122074-122091, 2022, doi: 10.1109/ACCESS.2022.3223099.

[13] F. Leo, E. Cocchi, y L. Brayda, «The effect of programmable tactile displays on spatial learning skills in children and adolescents of different visual disability», *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, n.º 7, pp. 861-872, jul. 2017, doi: 10.1109/TNSRE.2016.2619742.

[14] P. Mejía, L. C. Martini, F. Grijalva, J. C. Larco, y J. C. Rodríguez, «A survey on mathematical software tools for visually impaired persons: a practical perspective», *IEEE Access*, vol. 9, pp. 66929-66947, 2021, doi: 10.1109/ACCESS.2021.3076306.

[15] S. Gatto, O. Gaggi, L. Grosset, y L. G. N. Fovino, «Accessible mathematics: Representation of functions through sound and touch», *IEEE Access*, vol. 12, pp. 121552-121569, 2024, doi: 10.1109/ACCESS.2024.3448509.

[16] R. Jafri, S. M. M. Althbiti, N. A. A. Alattas, A. A. A. Albraiki, y S. H. A. Almuhawwis, «Tac-trace: a tangible user interface-based solution for teaching shape concepts to visually impaired children», *IEEE Access*, vol. 10, pp. 131153-131165, 2022, doi: 10.1109/ACCESS.2022.3228455.

[17] T. Tanabe, K. Nunokawa, K. Doi, y S. Ino, «Training system for white cane technique using illusory pulling cues induced by asymmetric vibrations», *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 30, pp. 305-313, 2022, doi: 10.1109/TNSRE.2022.3148770.

[18] G. V. Helden, V. Van Der Werf, G. N. Saunders-Smits, y M. M. Specht, «The use of digital peer assessment in higher education—an umbrella review of literature», *IEEE Access*, vol. 11, pp. 22948-22960, 2023, doi: 10.1109/ACCESS.2023.3252914.

[19] K. Villalba *et al.*, «Eyeland: a visually-impaired accessible english learning application using a design-based research framework», *IEEE Access*, vol. 12, pp. 142275-142290, 2024, doi: 10.1109/ACCESS.2024.3444741.

[20] M. N. Islam *et al.*, «A multilingual handwriting learning system for visually impaired people», *IEEE Access*, vol. 12, pp. 10521-10534, 2024, doi: 10.1109/ACCESS.2024.3353781.

[21] P. F. Balestrucci, E. Di Nuovo, M. Sanguinetti, L. Anselma, C. Bernareggi, y A. Mazzei, «An educational dialogue system for visually impaired people», *IEEE Access*, vol. 12, pp. 150502-150519, 2024, doi: 10.1109/ACCESS.2024.3479883.

[22] T. C. Lethbridge, S. E. Sim, y J. Singer, «Studying Software Engineers: Data Collection Techniques for Software Field Studies», *Empir Software Eng*, vol. 10, n.º 3, pp. 311-341, jul. 2005, doi: 10.1007/s10664-005-1290-x.

[23] O. Cico, L. Jaccheri, A. Nguyen-Duc, y H. Zhang, «Exploring the intersection between software industry and Software Engineering education - A systematic mapping of Software Engineering Trends», *Journal of Systems and Software*, vol. 172, p. 110736, feb. 2021, doi: 10.1016/j.jss.2020.110736.

[24] Object Management Group, «OMG Unified Modeling Language (OMG UML), Version 2.5.1», Object Management Group (OMG), Specification formal/17-12-05, dic. 2017. [En línea]. Disponible en: https://www.omg.org/spec/UML/2.5.1

[25] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3.ª ed. USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[26] T.-S. Nguyen, S. Stueker, y A. Waibel, «Super-Human Performance in Online Low-latency Recognition of Conversational Speech», 26 de julio de 2021, *arXiv*: arXiv:2010.03449. doi: 10.48550/arXiv.2010.03449.

[27] «Software Architecture in Practice, 3rd Edition». Accedido: 27 de julio de 2025. [En línea]. Disponible en: https://www.sei.cmu.edu/library/software-architecture-in-practice-third-edition/

[28] S. Geng, M. Josifoski, M. Peyrard, y R. West, «Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning», 18 de enero de 2024, *arXiv*: arXiv:2305.13971. doi: 10.48550/arXiv.2305.13971.

[29] J. Nielsen, *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994.