

A Hybrid Light-AI-based Framework for Accessibility Validation

Nicola Leonardi*
HIIS Laboratory, CNR – ISTI
Pisa, Italy
University of Pisa
Pisa, Italy
nicola.leonardi@isti.cnr.it

Marco Manca
HIIS Laboratory, CNR – ISTI
Pisa, Italy
marco.manca@isti.cnr.it

Fabio Paterno
HIIS Laboratory, CNR – ISTI
Pisa, Italy
fabio.paterno@isti.cnr.it

Abstract

Automated accessibility evaluation systems increasingly leverage large language models (LLMs) to assess conformance with established standards like the Web Content Accessibility Guidelines (WCAG). However, a fundamental methodological tension exists between the probabilistic, generalization-based nature of LLM inference and the deterministic, precisely scoped requirements of normative accessibility standards. A trustworthy, transparent, and accountable accessibility validation framework must be grounded in three principles: normative anchoring of evaluation logic to official guideline definitions, deterministic and traceable execution of assessment checks, and the systematic production of verifiable evidential artifacts. To operationalize these principles, we propose a computationally lightweight first, hybrid methodological framework for accessibility validation. The framework enforces a logical structure indicating the use of AI components, including small language models (SLMs), small multimodal models (SMMs), vision models, graph representations, and web agents, when rule-based techniques are insufficient to reliably evaluate criteria requiring human-like judgment and contextual understanding. We demonstrate the utility of this methodology, highlighting how it ensures algorithmic transparency and auditability, through concrete case studies.

CCS Concepts

• **Human-centered computing** → Accessibility; Accessibility design and evaluation methods; Accessibility; Accessibility systems and tools.

Keywords

Accessibility, LLM, SLM, Web Agents

ACM Reference Format:

Nicola Leonardi, Marco Manca, and Fabio Paterno. 2026. A Hybrid Light-AI-based Framework for Accessibility Validation. In *International Conference on Information Technology for Social Good (GoodIT '26)*, September 02–04, 2026, Pisa, Italy. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3794786.3830785>

*Corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License. *GoodIT '26, Pisa, Italy*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2483-1/2026/09
<https://doi.org/10.1145/3794786.3830785>

1 Introduction

The automation of web accessibility auditing has emerged as a significant challenge, driven both by the increasing complexity of digital interfaces and by growing legal obligations under regulations such as the European Accessibility Act and the Americans with Disabilities Act. Automated tools range from rule-based static analyzers to, more recently, systems that incorporate natural language understanding and vision-language models to detect accessibility barriers that cannot be detected through purely syntactic analysis. Recent contributions indicate that modern large language models can be used to automate the evaluation of specific accessibility techniques, especially in areas such as natural language understanding, semantic structure assessment, or the interpretation of alternative text. However, their role should not be seen as the sole solution within a framework that prioritizes professional auditing practice, alongside the traceability and reproducibility of results.

The capabilities of LLM-based evaluation must be considered within a broader methodological context and systematically compared to established, far less costly, deterministic, and reliable alternatives. Rule-based static analysers, DOM inspection tools, and computationally inexpensive heuristics that are grounded directly in normative guideline definitions have clearly defined evaluation failures. They produce stable outputs across repeated executions on identical inputs and generate findings that can be unambiguously traced to specific rule software implementations. In contrast, LLM-based evaluation introduces inherent sources of variability. These include sensitivity to prompt formulation, non-deterministic sampling behaviour, version-dependent model drift, and the lack of a transparent reasoning chain linking a finding to its normative basis. These characteristics conflict with the reproducibility and traceability requirements that distinguish professional auditing from informal heuristic review.

The appropriate research question, therefore, is not whether LLMs can contribute to accessibility validation, as the evidence suggests they can within certain bounded domains [1], [2], [3], but rather where and under what conditions their use provides meaningful and measurable improvement over standard rule-based approaches, and in what ways heterogeneous evaluation strategies can be systematically integrated to produce a new generation of validation tools. Such integration extends beyond the combination of LLM-based reasoning and deterministic rule engines. It encompasses computer vision techniques capable of detecting visually manifested accessibility barriers that cannot be detected by purely structural analysis, as well as web agent tools that can simulate user interaction sequences, navigate dynamic interface states, and discover accessibility failures that only emerge at runtime or in response to specific behavioural patterns. The central challenges

are thus determining which class of method is best suited to which category of accessibility check, how the outputs of heterogeneous components can be reconciled into a unified, traceable assessment record, and how the non-deterministic characteristics of AI-based evaluators can be bounded and made accountable within a framework that continues to meet the auditability standards required by professional practice. To address these challenges, this work investigates two central research questions:

RQ1: How can heterogeneous accessibility evaluation methods be systematically orchestrated within a unified framework while preserving normative traceability, reproducibility, and auditability requirements?

RQ2: What architectural principles should guide the integration of AI-based methodologies within accessibility validation pipelines in order to balance semantic capability, computational efficiency, and operational reliability?

To answer these questions, we propose a deterministic-first methodological framework for accessibility validation based on orchestration across rule-based analysis, lightweight semantic reasoning models, multimodal AI systems, graph-based structural representations, and automated interaction agents. The solution introduces a specific accessibility criterion-oriented execution strategy in which AI components are selectively activated only when deterministic techniques are insufficient.

2 Related Work

Traditional automated accessibility evaluation has relied on rule-based approaches that translate WCAG success criteria into deterministic checks. Tools such as WAVE, MAUVE++ [4], Siteimprove, and QualWeb [5] enable accessibility auditing by detecting syntactic and structural violations, including missing labels, invalid ARIA attributes, insufficient contrast ratios, and improper semantic markup. However, rule-based evaluators remain fundamentally limited in their ability to detect accessibility barriers requiring semantic interpretation, contextual understanding, or behavioural reasoning, leading to significant blind spots in the evaluation of many WCAG success criteria [2].

Recent research has increasingly explored the use of Large Language Models (LLMs) to extend accessibility validation beyond the capabilities of traditional automated checkers. Early studies primarily investigated prompt-based remediation and validation workflows. Othman et al. [6] employed GPT to automatically generate corrected HTML snippets from accessibility violations detected through the WAVE, while Delnevo et al. [7] explored LLM accessibility evaluation on HTML forms, tables, and images through prompt-based reasoning on isolated code snippets. Subsequent work progressively moved toward more structured AI-assisted accessibility pipelines. López-Gil and Pereira [2] proposed one of the earliest attempts to automate traditionally manual WCAG evaluations through LLM orchestration of image captioning, semantic similarity estimation, webpage summarization, and language analysis. Similarly, GenA11y [8] introduced criterion-specific extraction pipelines and contextual prompting strategies to evaluate a broader set of WCAG success criteria, while AccessGuru [9] proposed a taxonomy explicitly separating syntactic, semantic, and layout accessibility violations.

In parallel, several studies focused on limited accessibility sub-domains requiring specialized semantic or multimodal reasoning capabilities. Duarte et al. [10] explored the use of multiple LLMs for heading-related accessibility barriers. Pedemonte et al. [11] investigated multimodal LLM-based image-captioning systems to improve the accessibility of complex STEM-related web images. Calò et al. [12] conducted a study confirming the potentialities of LLMs in performing semantic analysis of Web content.

A particularly relevant research direction concerns the adoption of lightweight and specialized language models for accessibility-related semantic analysis. Kuzikov et al. [13] demonstrated that fine-tuned small language models can effectively perform semantic validation of WCAG 2.5.3 (“Label in Name”) with significantly lower computational costs than large proprietary LLMs, while maintaining acceptable classification performance.

More broadly, several recent works have explicitly acknowledged the limitations of current LLM-centric accessibility architectures. Paternò et al. [3] and Bassi et al. [14] emphasized the necessity of human oversight, hybrid evaluation strategies, and multi-step verification pipelines to mitigate hallucinations and improve accessibility auditing reliability. Aljedaani and Mollik [1] in a recent systematic literature review of LLM-supported web accessibility research, further identified recurring limitations across the field, including excessive dependence on prompt-based interaction strategies, relatively simple system architectures with a single LLM, weak integration of cognitive accessibility considerations, and limited orchestration frameworks capable of combining heterogeneous accessibility evaluation methods.

Thus, accessibility validation cannot be reliably reduced either to purely deterministic rule-based checking or to unrestricted general-purpose LLM reasoning. Existing work progressively converges toward the need for hybrid validation strategies. However, current approaches still largely lack explicit execution logic, selective AI activation policies, and computational governance strategies. This work addresses this gap by proposing an orchestration framework in which heterogeneous validation mechanisms are selectively combined according to the semantic and computational requirements of each WCAG success criterion.

3 The Hybrid Strategy Framework

The proposed architecture consists of heterogeneous components organized into a stack of execution layers. Before detailing the overall structure, each constituent element is described.

A core element of the framework is the **Context Builder**, whose role is to construct a focused, success-criterion-specific informational context to be passed to the following more advanced components, directing the analysis toward the precise subset of data relevant to the evaluation task at hand. The Context Builder operates at the level of structured DOM inspection and static content extraction, and its extraction logic is defined independently for each success criterion. Since it does not return any validation result on its own, it can be defined as a meta-component, to distinguish it from modules that independently produce standalone results.

The Context Builder fulfils an essential architectural function within the system. By filtering the informational input supplied to

the language or multimodal model, it serves multiple interdependent purposes: it reduces noise by excluding page content that is irrelevant to the criterion under evaluation; it limits prompt length, contributing to lower inference latency and reduced computational cost; it focuses the model's generative attention on the evidence that is normatively pertinent to the specific check being performed; and it mitigates the risk of hallucination by narrowing the solution space within which the model must reason, reducing the likelihood that the model will draw on unanchored prior knowledge rather than the concrete, extracted evidence provided. In this way, the Context Builder acts as a disciplined intermediary between the raw complexity of a web document, and the constrained, evidence-grounded reasoning that professional-grade accessibility evaluation requires.

The first mandatory component is the **Rule-Based DOM Inspector**, which operates as a deterministic, rule-driven, syntactic conformance evaluator that explicitly encodes each check as a programmatic instruction targeting specific DOM attributes, element properties, bounding boxes, or computed styles. A key characteristic is determinism and repeatability given an identical DOM snapshot, it produces identical results across executions, a property valuable for regression testing and audit reproducibility. The component lacks semantic understanding; it can reliably verify the presence of attributes (e.g., role, aria-label, alt, tabindex) and measure geometric properties such as bounding box dimensions or contrast ratios via computed CSS, but it cannot evaluate whether those attributes are meaningful or correct in context. Speed and scalability are notable strengths: because no model inference is involved, rule-based checks execute rapidly across large page surfaces and integrate readily into CI/CD pipelines or in Large Scale Evaluation process [15]. This component reflects the evaluation approach implemented by traditional rule-based tools such as Axe, Lighthouse, and Mauve++.

Three additional component categories contribute to the evaluation pipeline at different levels of perceptual and reasoning capability, Small Language Models (SLM), Small Multimodal Model (SMM) and Vision Model. A **Small Language Model**, like Phi-3-mini (Abdin et al., 2024) or Granite 4.1 3B (IBM Research, 2026), is a compact, text-only language model capable of performing semantic reasoning, classification, and rule interpretation over textual representations of specific part of the DOM or accessibility tree. SLMs in this framework are conceived as task-specialized semantic evaluators optimized for specific scoped accessibility reasoning tasks through criterion-oriented procedures. While lacking the broad generalization capacity of a foundation LLM, SLM offers substantially lower computational cost and can be deployed locally, making it suitable for the targeted task. Due to the probabilistic nature of the model, non-determinism behaviour and hallucinations are possible. Effective prompt design and proper context building are key factors in producing reliable outputs.

A **Small Multimodal Model** extends SLM capability by accepting text, image, and audio as simultaneous input modalities. Models in this category, such as Gemma4 [16], are designed to reason across heterogeneous input streams while remaining computationally efficient relative to large-scale multimodal systems. Unlike pure vision models, which process only pixel-level features,

an SMM jointly reasons over rendered visuals, DOM-derived representations, and textual context, enabling semantic alignment across heterogeneous modalities. In the context of accessibility evaluation, an SMM can for example correlate visual page rendering with its underlying DOM structure or reason about perceivable properties that are unmanageable to purely text-based analysis. SMM suffers the same issues of SLM regarding the non-deterministic generation, but computational cost and latency are correspondingly higher than those of only textual counterpart, so accurate management of the submitted task is essential to preserve practical applicability within the pipeline.

A **Vision Model** operates at the pixel and region level rather than at the semantic reasoning level. Models in this category, such as YOLO-based document object detectors [17], semantic segmentation architectures [18], or pixel-level analysis methods [19], process the rendered page as an image and identify structural elements, spatial regions, and visual boundaries. Within the framework, the output of the vision model is primarily intended to support higher-level reasoning components, such as SMMs or interaction agents; it serves as a perceptual grounding layer, providing precise geometric and layout information that higher-level components can subsequently interpret. Unlike language-based models, vision models do not reason about meaning; they provide a structured, spatially-aware decomposition of the visual surface, which is a necessary complement to DOM-derived structural information in cases where visual rendering diverges from the underlying markup status. Vision models are largely deterministic: given the same input image, they consistently produce equivalent detections and segmentations, making their output reliable and reproducible across evaluation runs. The computational costs and execution latency are relatively modest, making vision models well-suited for high-frequency, page-level visual analysis within an automated accessibility evaluation architecture.

As an illustrative example, consider the evaluation of WCAG criteria related to text overlap (Success Criterion 1.4.10 Reflow) during browser zoom and reflow operations. A rule-based DOM inspector may correctly report that all required elements are present and syntactically valid, while failing to detect that, in the rendered interface, portions of text become visually occluded.

A **Graph Model** component in the most basic configuration, can operate directly on the node-and-edge representations extracted directly from the DOM and the Accessibility Tree. The two representations are complementary, where the DOM provides structural completeness and layout fidelity, while the Accessibility Tree provides a semantically curated view of the interface as experienced by users of assistive technologies. Both representations are inherently hierarchical, but accessibility evaluation frequently requires reasoning about relationships that extend beyond parent-child containment. In accessibility evaluation, graph model representation is relevant because many WCAG success criteria depend on relational properties emerging across multiple interface components, rather than on the isolated correctness of individual DOM elements. It is suitable to perform tasks where relational and topological understanding are required: identifying cycles or inconsistencies in focus order (Success Criterion 2.4.3) and assessing the coherence of heading hierarchy as a navigational structure (S.C. 1.3.1) are examples of relationships that are either implicit, cross-branch, or

dynamically computed, and that a purely syntactic approach cannot efficiently capture. Beyond single-view graph representations, the graph-based methodological approach further enables the construction of heterogeneous graphs, in which nodes and edges may be of fundamentally different natures yet coexist within a unified representational structure, thereby affording a holistic and multifaceted understanding of the diverse entities and relationships that characterize a user interfaces. This representational heterogeneity allows the graph to simultaneously capture and integrate multiple complementary views and enables queries and inference patterns that are either cumbersome or intractable by other analysis approaches. In terms of computational characteristics, graph models are predominantly deterministic in a manner analogous to the rule-based inspector though operating at a relational rather than attribute level. Computational cost and latency are generally moderate and scale with graph size, which for typical web documents remains tractable, making the graph model a reliable and efficient complement to the reasoning-heavy components in the pipeline.

Finally, an **AI Web Agent** is the last required component for conducting a comprehensive accessibility assessment. This component is specifically designed to evaluate accessibility conditions that emerge only through dynamic interaction sequences and runtime behavioural flows, and that therefore cannot be reliably assessed through static DOM inspection or isolated semantic analysis alone. It integrates a Multimodal Language Model within a perception–reasoning–action loop: the agent observes the rendered page through screenshots, DOM serialization, or accessibility tree representations, reasons about what it observes, decides on a next action, executes it through a browser automation layer, and iterates. A key strength is the ability to adapt reasoning over novel UI patterns: because the agent reasons from general principles rather than a fixed ruleset, it can evaluate components it was never explicitly programmed to handle and explore alternative interaction paths toward the same goal. By simulating navigation from a user perspective, it can assess whether a form error message is contextually associated and understandable, or whether a focus order is logically coherent within the page’s information architecture. On the other hand, the component is subject to the inherent non-determinism of LM-based reasoning: the same page may yield slightly different findings across runs, and carefully designed prompts and contextual framing must be established in advance to mitigate this behaviour. Computational costs and execution latency are significant factors, more so than for other AI-based components, given the task’s recursive formulation, which limits practical application to targeted, high-value evaluation scenarios rather than exhaustive by default page-by-page scans.

Having introduced the individual components, their integration into a coherent, synergistic evaluation architecture can now be described.

To maximize execution efficiency and minimize computational overhead, the proposed framework defines a 3-tier preferential execution sequence for each criterion assessment. Not all tiers are necessarily invoked: simpler criteria, whose conformance can be determined through lightweight inspection, are resolved at earlier tiers without escalating to more resource-intensive stages:

Tier 1: Direct Code/DOM Extraction (Rule-based approach): Rapid, low-resource evaluation executable on standard CPU hardware. Extracts HTML attributes, text lengths, tag properties, and computed CSS features (e.g. measuring element widths, heights, and spacing directly) via DOM properties, without invoking language or vision models.

Tier 2: Specialized Structural and Semantic AI (SLM / Graph / Vision): Triggered when the task requires text comprehension (SLM) or complex relationship mappings (Graph modelling). This tier requires GPU support, though it remains operable on modest hardware configurations, such as a single GPU with 16 GB of VRAM.

Tier 3: Heavy Visual & Interactive Layers (SMM / Web Agent): Reserved as a last resort for criteria involving audio-visual understanding, dynamic interaction simulation, and navigation. Given the substantial computational demands of multimodal and agentic inference, its execution requires dedicated high-performance hardware, necessitating a minimum of 32 GB of VRAM and potentially a multi-GPU configuration.

To achieve a holistic approach to web accessibility validation, the proposed solution promotes combinations across tiers when a single method cannot satisfy all dimensions of a given success criterion. In such cases, the outputs of multiple components are reconciled into a unified, traceable assessment record that preserves full coverage of every aspect mandated by the normative evaluation requirements of the criterion under examination. By designing the framework around a rule-based-first workflow, the system aims to achieve enterprise-grade efficiency and scalability, proposing a disciplined and selective use of AI techniques strictly where they provide concrete, demonstrable support to lighter and more reliable deterministic approaches. Within this design philosophy, a deliberate preference for locally deployable small multimodal models over proprietary LLM API-based solutions represents not merely a pragmatic engineering choice but a principled architectural commitment. Small models offer substantial advantages in accessibility-sensitive evaluation contexts: they preserve data privacy by keeping all processing within a controlled infrastructure, reduce inference latency to levels compatible with iterative auditing workflows, eliminate dependency on external service availability and pricing policies, lower energy consumption in alignment with sustainability requirements, and afford a degree of transparency and controllability over model behaviour that opaque, remotely hosted systems cannot provide; moreover open-weight models can be fine-tuned to adapt their behavior to specific tasks or to enrich their knowledge of niche and specialized domains. The framework therefore treats the adoption of small, locally hosted multimodal models as a structural default, reserving recourse to more powerful external systems only in those narrowly scoped cases where local capabilities are demonstrably insufficient and where the associated traceability and reproducibility constraints can still be adequately satisfied.

At the architectural level, the proposed framework is designed to instantiate the strategy behavioral pattern [20] for carbon-aware services, in which progressively escalating computational tiers constitute the concrete strategies available for selection. Each tier is capable of delivering increasingly accurate and semantically rich evaluation outcomes, while incurring proportionally higher

computational and environmental costs. The dynamic selection among these strategies is governed by contextual signals, including the anticipated request volume and the forecast carbon intensity of the energy mix, ensuring that the most lightweight strategy capable of satisfying the evaluation requirements is preferentially invoked, and that computationally intensive tiers are engaged only when lower tiers prove insufficient to meet the required quality threshold.

4 End-to-End Pipeline Execution: A Case Study on WCAG Success Criterion 2.5.8

This section details the automated evaluation pipeline implemented for the decision logic defined by WCAG S.C 2.5.8 (Target Size – Minimum, Level AA) [21]. The goal of this criterion is to make controls easier to activate by ensuring that targets meet a minimum size or have sufficient spacing around them. The evaluation pipeline works across three viewport resolutions: 390×844 px (mobile representative), 768×1024 px (tablet/iPad portrait), and 1920×1080 px (desktop Full HD). These three resolutions were selected to capture the distinct layout breakpoints at which target-size failures most commonly manifest. For each resolution, the rule-based component first enumerates all interactive elements present in the rendered page by querying a curated CSS selector set that covers native HTML controls (button, input, select, textarea, a[href], option, and related form elements) as well as ARIA-role equivalents (role="button", role="link", role="tab", role="menuitem", etc). Each candidate element's bounding box (tag, XPath, width, height, top, left) is extracted and serialised to a per-resolution CSV file. The assessment then proceeds through five sequentially applied checkpoints, illustrated in Figure 1. The first checkpoint is a size check, which determines whether the element's rendered dimensions fall below the 24×24 CSS-pixel threshold; elements meeting or exceeding this threshold are immediately classified as passing. Elements that fail the size check are forwarded to an inline-text control check: controls that form part of a running sentence or paragraph (such as inline hyperlinks) are exempt from the spacing requirement and therefore also pass. Remaining elements are subject to the spacing offset check, which operationalises the WCAG 2.5.8 offset criterion by verifying that a 24 CSS-pixel-diameter circle centred on each undersized target does not intersect any other interactive target or the spacing circle of another undersized target; elements satisfying this condition pass. Elements that fail the spacing check are then evaluated by a semantic essentiality check: if the size and spacing of the targets is fundamental to the information being conveyed, the element is classified as essential and therefore passes. Finally, elements that survive all four gates are assessed by the AI-Web-Agent for the presence of an equivalent control, defined as a different element on the same page that triggers the identical action or function and itself meets the minimum 24 px size requirement; the presence of such an equivalent results in a pass. Only elements that fail every gate are reported as authentic violations of SC 2.5.8, with their tag name, XPath, actual dimensions, and annotated viewport-scrolling screenshots (overlaid with the diagnostic spacing circles) recorded as evidence.

The SC 2.5.8 was selected because it provides a compact stress test for the proposed orchestration strategy. Its normative evaluation combines deterministic geometric checks, concerning target size

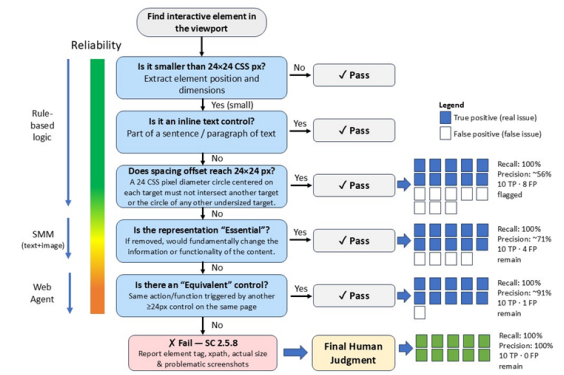


Figure 1: WCAG SC2.5.8 check flow implementation. The precision and recall calculations are for illustrative purposes only.

and spacing, with the essential and equivalent exceptions, which respectively require contextual interpretation and runtime functional exploration.

It is worth noting that the proposed methodological approach produces results at varying levels of precision and analytical depth. The first rule-based component, which carries the lowest computational overhead, is alone capable of returning a set of potentially non-compliant elements: all interactive elements with dimensions below 24×24px that are not inline and are spaced too closely to one another. The subsequent multimodal language model and web agent components then progressively filter this initial candidate list. This sequential architecture reflects a deliberate design choice: optimizing for recall over precision by tolerating potential false positives. This trade-off is both rational and well-motivated in the accessibility validation context, where the two types of error carry an inherently asymmetric cost. A false negative, meaning a real accessibility issue that goes undetected, may have serious consequences for users with disabilities, whereas a false positive merely requires a human reviewer to manually discard an incorrectly flagged element. Given this asymmetry, maximizing recall at the expense of precision is the appropriate strategy, with the understanding that in the more complex cases requiring comprehensive contextual understanding, the final determination rests with a human reviewer. Figure 1 shows that recall remains ideally at 100% throughout the process; in fact, no real problem is ever discarded. Each component progressively removes false positives, improving accuracy while the human reviewer handles any remaining edge cases.

Although specific to SC2.5.8, the figure highlights another core design principle of the approach: the deliberate confinement of uncertainty and probabilistic reasoning — and the attendant risk of hallucination — inherent to the use of multimodal language models and the web agents that internally rely on them, to the later stages of the pipeline. This principle is visually conveyed through a gradient colour bar ranging from green to orange, encoding the concept of reliability. We argue that it is appropriate for the system to surface this information about the nature of the outputs produced at each pipeline stage, thereby enhancing result transparency and

Table 1: Small interactive elements detected by Claude Sonnet 4.6 (using the right screenshot of Figure 2)

Detected elements	Estimated size	Result
Day cells (contribution graph)	10 × 10 px	True Positive
more-options (...) button	20 × 20 px	True Positive
Learn how we count	14 px tall	True Positive
contr. link		
Repo links	16 px tall	True Positive
Org avatar icons	20 × 20 px	False Positive
Nav tab labels	22 px tall	False Positive
Repo stars and forks links	Not detected	False Negative
Block or report user link	Not detected	False Negative

enabling human reviewers to more rapidly and effectively direct their attention toward the least certain and most ambiguous aspects of their evaluations.

4.1 A practical Example: GitHub page

To illustrate the framework with a real-world instance, we apply it to the personal GitHub profile of a prominent AI communicator and research engineer [22].

Before presenting the full pipeline in operation, it is considered necessary to first check the results by performing the validation entirely through a multimodal model. Two multimodal models were evaluated for this purpose: a large proprietary model, Claude Sonnet 4.6 [23], and a small open-weight model, Gemma 4B E2B [16]. A dedicated prompt (provided in section A of the supplementary materials¹) was designed to extract the potential interactive targets on which the criterion is to be verified. A screenshot of the HTML rendered page was provided as input, and, to facilitate the task, the resolution of the screenshot was explicitly declared within the prompt itself.

The results reported in Table 1 demonstrate that direct application of a multimodal model proves insufficient for accurately determining the dimensions of interactive elements from visual input alone, even when using large-scale architectures (similar results were obtained with the Gemma 4B E2B model).

The model consistently produces both false positives and false negatives, thereby rendering this deployment paradigm ineffective for the specific success criteria assessment.

Having demonstrated that the direct and exclusive application of a multimodal model to this success criterion verification does not represent an optimal approach in standard real situations, now we focus on the integration of such a model as a downstream component, operating after the processing performed by the lightweight rule-based module.

Recalling Figure 1, the rule-based component can identify all interactive elements whose dimensions are less than 24x24 pixels, are not inline, and do not maintain sufficient spacing from one another. The component therefore returns its list of potentially

non-compliant elements with respect to WCAG guidelines. However, by its nature, it cannot proceed further in the verification process to assess the additional exceptions expressed in the guideline, namely the Equivalent and Essential ones, and consequently cannot exclude from the initial list those elements that fall into these exception categories. This kind of exception check is delegated to the advanced components introduced previously, namely a Small Multimodal Model and a Web agent.

Indeed, determining whether a particular element should be considered “Essential” requires a thorough understanding of its context and role within the actual user interaction. This information can be conveyed through a visual screenshot of the global interface view, which motivates the adoption of a Multimodal Model for this specific subtask. By providing the model with the interface image as contextual input alongside an annotation of the potentially non-compliant elements identified in the preceding step (rule-based output), the model is potentially capable of assessing the essentiality of each flagged element.

“Equivalence”, on the other hand, must be evaluated on functional grounds rather than on structural or markup characteristics alone. A rigorous assessment of equivalence therefore necessitates verification across three dimensions: *destination*, ensuring that the alternative control leads to the same content or outcome; *modality*, confirming the absence of any functional difference in the manner in which the interaction is performed; *availability*, verifying that the alternative control is present within the same context or viewport. It follows that equivalence represents a functional judgment rather than a syntactic or structural comparison, a distinction that carries significant implications for automated verification approaches. For the reasons outlined above, the verification of equivalence has typically required human judgment [21]. However, the remarkable performance improvements recently exhibited by AI-enabled web agents across a range of benchmarks [24] suggest their potential adoption for this specific task.

The processing steps and corresponding results are presented below. The lightweight rule-based component is able to identify targets smaller than 24x24 CSS pixels that do not fall under the inline exception and that fail to satisfy the spacing requirement. In addition to providing a structured (.csv) or semi-structured (e.g. JSON) dump of the flagged elements, the component is also capable of generating screenshots, corresponding to sequential scrolls of the page, onto which annotations are overlaid to assist the subsequent analysis components. In this specific implementation, circular annotations with a diameter of 24 pixels were adopted, directly operationalizing the verification procedure described in success criterion SC 2.5.8. These annotated screenshots can also be incorporated into the report generated as output by the system, serving a dual purpose. On one hand, they provide concrete documentary evidence to support accessibility auditing activities, and on the other hand, they ensure the traceability of results, reinforcing accountability and the long-term reliability of the evaluation process.

The screenshot illustrated in Figure 2 displays several elements annotated with red circles. While some of these circles are sufficiently spaced to avoid mutual intersection and therefore fall under the spacing exception, certain regions exhibit overlapping annotations, as is particularly evident in the heatmap representing the

¹Supplementary material, including the prompts used, will be made available upon request.

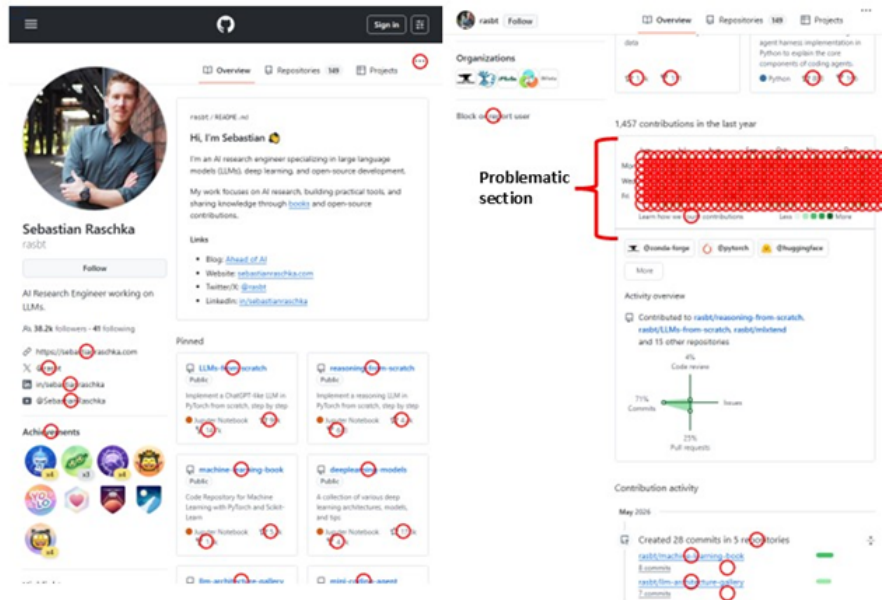


Figure 2: Page screenshots with small ($< 24 \times 24$ px) targets marked; the densest region of intersecting targets is manually outlined for clarity

author’s contribution history over the course of the year. To assess whether this section qualifies for the “Essential” exception, the annotated screenshot is submitted to the multimodal model alongside a dedicated prompt (supplementary materials section B) that specifies the nature of the circular annotations and the task to be performed. The model is thus relieved of the need to execute the extensive task of interactive element extraction and dimension computation, freeing it to direct its capabilities toward the visual identification of intersecting red circles and determine, given the overall contextual information conveyed by the screenshot, whether the component in question should be regarded as essential.

The following is an exemplary output generated by the Claude Sonnet 4.6 model (similar results were also obtained using Gemma4 E2B), in which the aforementioned elements are classified as non-essential and attributed purely to stylistic choice.

Model output:

```
[
  {
    "element name": "Contribution graph cells",
    "element description": "The dense grid of small squares in the '1,457 contributions in the last year' heatmap calendar",
    "is_essential": false,
    "reason": "While the calendar uses a grid layout to represent days, the core information — contribution frequency over time — could be conveyed equally well with larger cells. The size is a design decision to fit a full year compactly on screen, not a legal requirement or a case where resizing would fundamentally destroy the meaning. No Essential exception applies."
  }
]
```

Given that this pipeline component returns that the elements annotated with intersecting red circles do not fulfil an essential role, the subsequent step, prior to their definitive association with failure elements of the criterion, consists in verifying the final check on the presence or absence of a functionally equivalent element. For this task, the Web Agent tool browser-use [25] was employed with the multimodal model GPT-4o as engine. The agent runs a continuous perception–reasoning–action loop where on each step it does two things simultaneously to understand the page: it extracts a compressed DOM tree (the structured HTML, with Bounding Box filtering and paint-order pruning applied) and takes a screenshot. Both elements are sent to the LLM. In this specific case, the adoption of a proprietary Language Model has been dictated by the complexity of the task and the amount of information to manage.

The component is triggered to follow the task described in section C of the supplementary materials, where the agent’s input parameters are not hardcoded but are derived directly from the field extracted in the previous step. As an illustrative example, a four steps run is reported:

- Task request: Open the URL <https://github.com/rasbt> and check if the page offers an ‘Equivalent’ function to the ‘Contribution graph cells’... - Answer YES or NO and justify your answer.
- Steps 1–4 (automated): The agent executed 4 exploratory steps examining the heatmap and surrounding controls before concluding.
- Out: “NO, the GitHub profile page does not offer an equivalent function to the ‘Contribution graph cells’ that meets WCAG Success Criterion 2.5.8 Target Size (Minimum). The

dense grid of small squares in the heatmap calendar does not have an alternative control on the same page that provides equivalent functionality with larger target sizes or improved accessibility.”

During its steps, the agent performs actions of varying nature: searching for “Contribution graph cells” text across the page DOM, scrolling through the page, clicking on elements of the interactive heatmap and on buttons identified as potential equivalents, and comparing the resulting output, demonstrating good flexibility and a diversified repertoire of actions in pursuing the main task. Ultimately, the element is classified as having no equivalent and therefore as a non-compliant for the criterion.

Upon completion of the processing pipeline across the various framework modules, the subset of elements constituting the contribution gridmap was classified as non-essential and lacking an equivalent control; consequently, it was correctly reported as a genuine accessibility issue.

Although the proof of concept presented in this work focuses on SC 2.5.8, the same orchestration strategy naturally extends to many other WCAG success criteria by activating different subsets of the proposed components according to the evidence required by the criterion.

5 Conclusions and Future Work

The proposed accessibility validation framework is grounded in the premise that high-stakes domains such as accessibility compliance, where failures can expose organizations to significant legal liability, are fundamentally ill-suited to a “black box” AI-first approach. In such contexts, three properties are non-negotiable: transparency, meaning that the rationale behind every decision must be traceable and intelligible to human reviewers, accountability, meaning that the specific procedure responsible for a given verdict must be identifiable and auditable, and trustworthiness, meaning that the system guarantees reproducibility, ensuring identical digital assets produce identical compliance scores every time. Relying exclusively on probabilistic models would undermine these requirements while introducing operational and financial costs that render such an approach impractical for most organizational contexts.

To address these constraints, the proposed paradigm decomposes the verification of each success criterion into a sequence of processing layers arranged by increasing complexity and operational cost. A more computationally intensive layer is engaged only when the criterion verification requires advanced semantic understanding or a higher degree of accuracy is required to reduce false positives that lighter preceding layers cannot adequately resolve, ensuring that resources are consumed proportionally to the actual difficulty of each verification task.

Reserving expert human judgment only for genuinely complex edge cases, it effectively scales the volume of accessibility verification work that would otherwise be unmanageable through purely manual processes, without sacrificing the transparency and accountability that compliance contexts demand.

This study, starting from the analysis and critical awareness of the automated validation tools currently available for accessibility criteria assessment, highlighted their intrinsic limitations. It

then examined how several recent studies have attempted to apply artificial intelligence techniques, particularly proprietary Large Language Multimodal Models, yielding interesting results that are often appealing in terms of implementation simplicity. However, such studies do not address the problem holistically, as they fail to consider the broader spectrum of relevant dimensions, including economic impact, environmental sustainability, reliability, accountability, and privacy implications entailed by the adoption of such solutions.

Both initial research questions have been addressed, paving the way for a new generation of automated validation tools that do not diminish the value of classical rule-based methodologies but rather augment them with the latest artificial intelligence techniques, achieving a balanced and constructive integration between the two approaches.

The framework defines a general architectural backbone and the underlying principles governing its design, demonstrating the potential of the methodology through a case study focused on success criterion 2.5.8. Nevertheless, several open questions remain, among which the most prominent is the need for empirical validation supported by systematic experimentation and quantitative metrics across a broader range of criteria and real-world case studies.

Furthermore, the predefined sequential adoption of specific methods may prove rigid in certain success criteria check. A promising direction therefore consists of introducing an intelligent orchestrator capable of selecting components not through deterministic programmatic logic but based on the effective contribution each module provides toward task resolution.

Acknowledgments

We acknowledge the use of Generative AI tools only to improve the grammar, style, and readability of this manuscript.

References

- [1] W. Aljedaani and R. H. Mollik, ‘Large Language Models for Web Accessibility: A Systematic Literature Review’, arXiv.org. Accessed: May 20, 2026. [Online]. Available: <https://arxiv.org/abs/2605.13873v1>
- [2] J.-M. López-Gil and J. Pereira, ‘Turning manual web accessibility success criteria into automatic: an LLM-based approach’, *Univ Access Inf Soc*, vol. 24, no. 1, pp. 837–852, Mar. 2025, doi: 10.1007/s10209-024-01108-z.
- [3] F. Paternò, M. Vinci, M. Manca, and N. Iannuzzi, ‘How an LLM Can Improve Automatic Web Accessibility Validation?’, in *CHIItaly ’25*. New York, NY, USA: ACM, Oct. 2025, pp. 1–8. doi: 10.1145/3750069.3750310.
- [4] G. Broccia, M. Manca, F. Paternò, and F. Pulina, ‘Flexible Automatic Support for Web Accessibility Validation’, *Proc. ACM Hum.-Comput. Interact.*, vol. 4, no. EICS, p. 83:1–83:24, Jun. 2020, doi: 10.1145/3397871.
- [5] N. Fernandes, R. Lopes, and L. Carriço, ‘On web accessibility evaluation environments’, in *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, in *W4A ’11*. New York, NY, USA: ACM, Mar. 2011, pp. 1–10. doi: 10.1145/1969289.1969295.
- [6] A. Othman, A. Dhoub, and A. Nasser Al Jabor, ‘Fostering websites accessibility: A case study on the use of the Large Language Models ChatGPT for automatic remediation’, in *PETRA ’23*. New York, NY, USA: ACM, Aug. 2023, pp. 707–713. doi: 10.1145/3594806.3596542.
- [7] G. Delnevo, M. Andruccioli, and S. Mirri, ‘On the Interaction with Large Language Models for Web Accessibility: Implications and Challenges’, in *2024 IEEE CCN*, pp. 1–6. doi: 10.1109/CCNC51664.2024.10454680.
- [8] Z. He, S. F. Huq, and S. Malek, ‘Enhancing Web Accessibility: Automated Detection of Issues with Generative AI’, *Proc. ACM Softw. Eng.*, vol. 2, no. FSE, p. FSE101:2264–FSE101:2287, Jun. 2025, doi: 10.1145/3729371.
- [9] N. Fathallah, D. Hernández, and S. Staab, ‘AccessGuru: Leveraging LLMs to Detect and Correct Web Accessibility Violations in HTML Code’, in *ASSETS ’25*. New York, NY, USA: ACM, Oct. 2025, pp. 1–22. doi: 10.1145/3663547.3746360.
- [10] C. Duarte, M. Costa, L. Seixas Pereira, and J. Guerreiro, ‘Expanding Automated Accessibility Evaluations: Leveraging Large Language Models for Heading-Related

- Barriers', in Companion Proceedings of the 30th International Conference on Intelligent User Interfaces, in IUI '25 Companion. New York, NY, USA: ACM, Mar. 2025, pp. 39–42. doi: 10.1145/3708557.3716329.
- [11] G. Pedemonte, M. Leotta, and M. Ribaud, 'Improving Web Accessibility With an LLM-Based Tool: A Preliminary Evaluation for STEM Images', IEEE Access, vol. 13, pp. 107566–107582, 2025, doi: 10.1109/ACCESS.2025.3577519.
- [12] T. Calò, A.-E. Gurita, and L. De Russis, 'Measuring the Semantic Accessibility Gap in LLM-Generated Web UIs', in CHI EA '26. New York, NY, USA: ACM, Apr. 2026, pp. 1–5. doi: 10.1145/3772363.3799364.
- [13] B. O. Kuzikov, O. A. Shovkoplias, P. O. Tytov, and S. R. Shovkoplias, 'Application of small language models for semantic analysis of Web interface accessibility', PROBLEMS IN PROGRAMMING, no. 2, pp. 77–86, Sep. 2025, doi: 10.15407/pp2025.02.077.
- [14] B. Bassi *et al.*, 'An Assessment of LLM-Based Auditing and Validation for Web Accessibility', in GoodIT '25. New York, NY, USA: ACM, Dec. 2025, pp. 297–305. doi: 10.1145/3748699.3749805.
- [15] N. Iannuzzi, M. Manca, F. Paternò, and C. Santoro, 'Large Scale Automatic Web Accessibility Validation', in GoodIT '23. New York, NY, USA: ACM, Sep. 2023, pp. 307–314. doi: 10.1145/3582515.3609549.
- [16] 'Gemma 4', Google DeepMind. Accessed: May 21, 2026. [Online]. Available: <https://deepmind.google/models/gemma/gemma-4/>
- [17] 'ultralytics/ultralytics: Ultralytics YOLO'. Accessed: May 21, 2026. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [18] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, 'Masked-attention Mask Transformer for Universal Image Segmentation', arXiv.org. Accessed: Jun. 01, 2026. [Online]. Available: <https://arxiv.org/abs/2112.01527v3>
- [19] Gonzalez, Rafael C., Digital Image Processing, Pearson. in 4th. 2018.
- [20] S. Forti, J. Soldani, and A. Brogi, 'Carbon-Aware Software Services', in ESOC 2025, Bolzano, Italy, February 20–21, 2025, Proceedings, Berlin, Heidelberg: Springer-Verlag, Feb. 2025, pp. 65–80. doi: 10.1007/978-3-031-84617-5_6.
- [21] W3C Web Accessibility Initiative (WAI), 'Understanding Success Criterion 2.5.8: Target Size (Minimum)'. [Online]. Available: <https://www.w3.org/WAI/WCAG22/Understanding/target-size-minimum.html>
- [22] 'rasbt - Overview', GitHub. Accessed: Jun. 01, 2026. [Online]. Available: <https://github.com/rasbt>
- [23] Anthropic, 'Claude Sonnet 4.6 System Card', Feb. 2026. [Online]. Available: <https://anthropic.com/claude-sonnet-4-6-system-card>
- [24] S. Zhou *et al.*, 'WebArena: A Realistic Web Environment for Building Autonomous Agents', Apr. 16, 2024, arXiv: arXiv:2307.13854. doi: 10.48550/arXiv.2307.13854.
- [25] browser-use/browser-use. (Jun. 01, 2026). Python. Browser Use. Accessed: Jun. 01, 2026. [Online]. Available: <https://github.com/browser-use/browser-use>