

Measuring AI-Agent Accessibility of Web Applications: A Construct-Development Framework

**Oluwaseyi Adediran¹, Odumuyiwa Teslim², Oluwafemi Memuletiwon³,
Oluwaseun Ogunseitan⁴, Joseph Adewuyi⁵, Alfred Udosen⁶, Opeyemi Adelowo⁷**

Faculty, Department of Information Technology^{1,7}

Student, Department of Information Technology^{2,3}

Staff, Division of Information Communication Technology⁴

Faculty, Department of Computer Science^{5,6}

Babcock University, Ilishan Nigeria

Abstract: Agents built on large language models (LLMs) now browse, fill in forms and complete transactions on websites that were designed for people looking at a screen. Existing standards do not describe what such an agent needs from a page. WCAG 2.2 sets requirements for human users of assistive technology, and interoperability standards assume two systems that already share a structured contract. Neither says when a web interface can be reliably perceived, acted on and recovered from by an autonomous agent. This paper proposes the Agent Accessibility Measurement Framework (AAMF) to fill that gap. AAMF divides the construct into Core In-Page Accessibility, made up of Structural Interpretability (SI), Semantic Richness (SR), Action Executability (AE) and Resilience and Error Recoverability (RR), and a separate Machine Discoverability and Protocol Support (MDP) pillar. We define 25 indicators with explicit formulas, an equal-weight baseline score, a failure-criticality-weighted composite score, and a five-layer Agent Interaction Stack (AIS) that keeps in-page interaction apart from out-of-band discovery. We place the construct within ISO/IEC 25010:2023, work through one hypothetical scoring example, and derive 25 engineering guidelines from the indicator definitions. The paper specifies the framework only. It reports no audit of production websites and no agent task study; Section VII describes the study that is needed to validate it.

Keywords: AI agents, web accessibility, accessibility tree, semantic HTML, agentic web, software quality models, WCAG, Model Context Protocol, measurement framework

I. INTRODUCTION

LLM-based agents can now read the structure of a page, plan several steps ahead, fill in forms, extract structured data and complete purchases on ordinary web applications without a person operating the browser [1], [2], [3]. Industry forecasts expect these agents to handle a growing share of online commerce and enterprise work within this decade [4]. The Web they operate on, however, was built for people. Pages are arranged for human eyes and human layout conventions, and they assume a reader who can tolerate some ambiguity [5]. Current front-end practice makes this worse. Single-page applications that hydrate on the client, deeply nested non-semantic containers, icon-only buttons and aggressive bot mitigation all get in the way of a machine navigator, and none of that friction depends on how capable the model is.

This helps explain a gap that appears again and again in the benchmark literature. On WebArena and Mind2Web, strong LLM agents still complete realistic multi-step tasks far less often than people do [2], [3]. The usual explanation puts the blame on the agent: weak planning, poor visual grounding, or context that runs out. That account is incomplete.



A benchmark score cannot tell an agent that reasoned badly apart from one that was shown three identical, unlabeled buttons inside a <div> with no semantic role. Task completion measures agent capability and environmental accessibility together, and the literature has almost no instrument for measuring the second on its own.

Human web accessibility faced a similar problem about thirty years ago. The response then was to write requirements for the environment itself, which became the Web Content Accessibility Guidelines (WCAG) [6], instead of waiting for assistive technology to get smarter. We think the same step is now due for autonomous agents. WCAG conformance gives a necessary semantic base, but it does not go far enough. WCAG was written around human cognition mediated by assistive technology. A screen-reader user who hears three buttons labeled "Edit" can use spatial context to work out which is which. An LLM agent reading a flattened serialization of the accessibility tree has no such context, and it pays for the ambiguity in tokens and retries. WCAG also says nothing about protocol-level discovery, that is, whether an agent can find a site's machine-readable manifests, schemas or transactional endpoints before it loads a single page.

We ask four research questions. RQ1: Can agent accessibility be defined as a software quality attribute that is distinct from accessibility for people with disabilities and from system interoperability? RQ2: Can the construct be broken down into indicators that have explicit formulas and can be computed from a page's markup and HTTP headers? RQ3: Can those indicators be designed so that failures with different architectural causes (missing landmarks, ambiguous action targets, absent structured data, absent machine manifests) stay visible separately instead of being averaged into one number? RQ4: What engineering guidance follows directly from the indicator definitions?

The paper makes four contributions. First, it gives a formal definition of agent accessibility, places it in the ISO/IEC 25010:2023 quality model [7], and introduces the five-layer Agent Interaction Stack (AIS), which separates in-page interaction from external discovery. Second, it presents the Agent Accessibility Measurement Framework (AAMF): five pillars, 25 indicators with explicit formulas, and two composite scores, one unweighted and one weighted by failure criticality. Third, it specifies how a static-DOM auditing tool computes the indicators and works through one hypothetical page from raw scores to composite. Fourth, it derives 25 engineering guidelines, one for each indicator. We make no empirical claims about real websites. Section VIII sets out what a sound audit would require, and we treat that audit as the next paper.

The rest of the paper is organized as follows. Section II reviews related work. Section III defines the construct and the AIS. Section IV specifies the indicators and scores. Section V lists the guidelines, Section VI discusses the framework's claims and predictions, Section VII covers limitations and the required empirical study, and Section VIII concludes.

II. RELATED WORK

A. Autonomous Web Agents and How They Read Pages

Early programmatic navigation relied on robotic process automation and headless testing tools such as Selenium, Puppeteer and Playwright, driven by brittle CSS or XPath selectors. LLMs changed what these tools could do. WebGPT showed that a model fine-tuned with human feedback could search, follow links and synthesize answers [8]. WebAgent paired a planning model with HTML sub-tree parsing to act on real forms [9]. ReAct introduced the Thought, Action, Observation loop, which interleaves reasoning with grounded actions [10], and AutoWebGLM combined HTML simplification with reinforcement learning to beat larger general-purpose models on web navigation benchmarks [5]. Multimodal systems, surveyed under the label of GUI agents and evaluated on VisualWebArena, add rendered screenshots to the DOM [11], [12].

These systems read a page in one of four ways. Some take the raw or pruned DOM, which keeps full syntactic detail at a high token cost. Some use the platform accessibility tree, which the browser computes from the DOM and CSSOM for assistive technology and which drops most visual noise [13], [14]. Some work from screenshots, which sidesteps DOM quirks but introduces coordinate errors and costs more tokens. Others combine an accessibility-tree serialization with targeted visual grounding. Nearly all of this work treats the host application as a fixed, somewhat hostile environment and puts its effort into better parsing, pruning and grounding on the agent side. We study the boundary itself. If the interface exposes clean structure, much of that agent-side effort is no longer needed.



B. Web Agent Benchmarks and the Attribution Problem

Mind2Web introduced more than 2,000 tasks across 137 real websites, with cross-site and cross-domain splits [3]. WebArena built a self-hosted set of working sites covering e-commerce, software development, knowledge management and forums [2]. VisualWebArena added multimodal reasoning over visually ambiguous interfaces [12]. ST-WebAgentBench proposed the Completion-under-Policy metric and showed that nominal task success can hide serious policy violations in enterprise settings [15]. AgentWebBench evaluated coordination between a user-facing agent and per-site content agents in a decentralized retrieval setup [16].

Each of these benchmarks asks where the agent's planner or grounder went wrong. None of them gives a standard way to ask the reverse question: how much of the failure came from the host application's markup, hydration behavior, unlabeled controls or unclear state changes? A task-completion rate mixes agent skill and environmental accessibility with no way to separate the two. AAMF is meant to supply the environment-side measurement.

C. WCAG Is Necessary but Not Sufficient

WCAG 2.2 is organized around four principles: Perceivable, Operable, Understandable and Robust [6]. Screen readers and DOM-based LLM agents consume the same accessibility tree [17], [13], [14], so it is tempting to assume that a WCAG-conformant site is also agent accessible. We argue that conformance is necessary but not sufficient, for three reasons that come from how the two kinds of reader differ.

The first reason is the cost of context. A screen-reader user can skip or speed through verbose output. An LLM agent has to ingest every serialized token of the DOM or accessibility tree, so verbose markup is a direct operating cost that grows with page size. The second is target disambiguation. A person can tell three "Edit" buttons apart by their position on screen, while an agent working from a flat serialization needs a unique programmatic handle for each one. The third is discovery. WCAG sets no requirements for machine-readable manifests, structured transactional endpoints or idempotent error semantics, and these matter far more to a system that has no human common sense to fall back on.

D. The Semantic Web and Machine-Readable Manifests

The semantic web vision [18] found a practical successor in Schema.org and JSON-LD [19], [20], and more recently in the Model Context Protocol (MCP) [21] and the community-proposed /llms.txt convention [22]. In practice, site operators add Schema.org markup mostly to win rich search results [23]. They rarely describe what a page can do operationally or transactionally, and research has mostly treated APIs and human-facing HTML as separate worlds. The MDP pillar of AAMF evaluates the layer that connects them, which existing frameworks leave out.

E. Comparative Positioning

Table I compares AAMF with the frameworks above and with WebQEM, an earlier web quality model built around human usability [24]. The comparison shows that no existing framework covers action disambiguation, out-of-band discovery and error recoverability together, and none produces a composite score aimed at autonomous agents.

TABLE I Comparison of AAMF with Existing Evaluation Frameworks

Framework	Primary target	In-page semantic structure	Action disambiguation	Out-of-band discovery	Error recoverability	Composite score
WebArena [2]	Autonomous agents	Indirect (DOM)	Treated as agent error	None	None	Task success rate only
Mind2Web [3]	Autonomous agents	Raw element trees	Target matching only	None	None	Step success rate only
WCAG 2.2 [6]	Humans with disabilities	High (POUR)	Human contextual judgment	None	Visual error text	Conformance levels A, AA, AAA



Schema.org [19]	Search engine crawlers	Entity metadata only	None	Sitemaps	None	None
WebQEM [24]	Human website users	Usability and layout	Usability inspection	General HTTP	Usability reliability	Weighted preference tree

III. THEORETICAL FOUNDATIONS

A. Construct Definition and Scope

Definition 1 (Agent Accessibility). Agent accessibility is a property of an interface and its environment. It describes how reliably, deterministically and efficiently an autonomous software agent can perceive application state, identify and disambiguate action targets, carry out multi-step workflows, and recover from exceptional states without out-of-band human help or perceptual intuition.

The definition separates two kinds of access. In-page interaction accessibility, measured by the SI, SR, AE and RR pillars, concerns whether an agent that has already loaded a page can perceive and act within it. Machine discoverability, measured by the MDP pillar, concerns whether an agent can locate resources and learn its permissions out of band, through files such as robots.txt, sitemap.xml or /llms.txt. The two vary independently. A site can have excellent in-page accessibility and no external discoverability, or the reverse.

Agent accessibility also differs from two neighboring ideas. Interoperability assumes that two systems already share a structured contract, such as an OpenAPI service. Agent accessibility covers the case where no such contract exists and the agent must work through an interface designed for human viewing. Automatability, meaning that a control can be triggered by a synthetic DOM event, is a yes-or-no scripting property. A control can be clickable from a script and still be semantically opaque to an agent that first has to find it, so automatability does not imply accessibility.

B. Placement in Software Quality Models

We treat agent accessibility as a characteristic that cuts across three parts of ISO/IEC 25010:2023 [7]. It extends Compatibility, specifically interoperability and co-existence, to autonomous software actors. It extends Interaction Capability (formerly Usability) by widening the "specified user" to include artificial agents. It also bears on Reliability, through fault tolerance and recoverability when the user interface changes unexpectedly.

C. The Agent Interaction Stack

The AIS describes five surfaces through which an agent can interact with a web application (Fig. 1). The layers complement each other; they are not a strict hierarchy, and a site can implement a higher layer while neglecting a lower one.

Layer 1, the Structural DOM, is the document hierarchy parsed from raw markup. Layer 2, the Accessibility Tree, exposes semantic roles, accessible names and live state updates with visual noise removed [13], [14]. Layer 3, the API and Transactional layer, covers direct programmatic operations and idempotent revalidation [25]. Layer 4, the Protocol and Discoverability layer, holds machine manifests and crawling directives that an agent reads before or outside any page visit. Layer 5, the Semantic and Ontological layer, supplies authoritative linked-data entities such as JSON-LD, so the agent does not have to infer what a page describes [20].



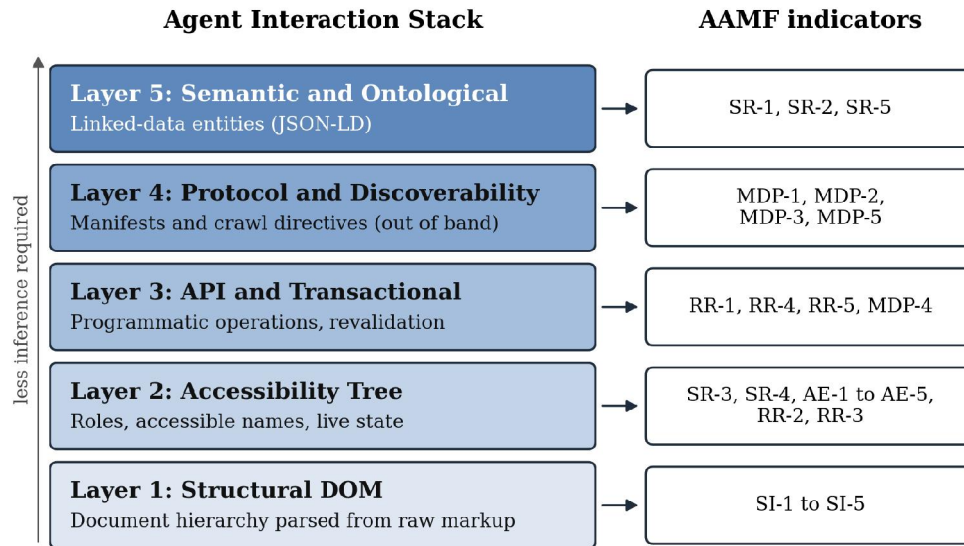


Fig. 1 The five layers of the Agent Interaction Stack and the AAMF indicators that measure each layer. An agent confined to Layer 1 has to spend its computational budget inferring purpose and outcome from raw syntax. Each higher layer an application implements reduces that burden, because the agent receives authoritative metadata and direct action bindings where it would otherwise have to guess. Fig. 1 also shows which AAMF indicators measure each layer, which ties the conceptual model in this section to the measurement instrument in the next.

IV. THE AGENT ACCESSIBILITY MEASUREMENT FRAMEWORK

AAMF measures agent readiness with 25 indicators grouped into five pillars. Four pillars (SI, SR, AE, RR) make up Core In-Page Accessibility and one (MDP) covers Machine Discoverability and Protocol Support. Every indicator lies in $[0, 1]$, where 1 is best. Unless noted otherwise, each can be computed from static markup and HTTP response headers. When the denominator of a ratio indicator is empty (for example, a page with no images for SR-3), the indicator scores 1 and is reported as not applicable; SI-1 is the one exception, as defined below.

A. Pillar 1: Structural Interpretability (SI)

SI-1, Functional Region Semantic Partitioning. The share of functional regions actually present on the page (such as top navigation, main content and footer) that sit inside a standard HTML5 landmark element or carry an explicit ARIA landmark role, as opposed to a generic container. The score is 0 if no functional region can be identified.

$$I_{SI-1} = \frac{|R_{\text{present}} \cap R_{\text{landmark}}|}{|R_{\text{present}}|}$$

Because the score is computed over regions that exist, a page is not penalized for leaving out an optional region such as `<aside>`.

SI-2, Heading Hierarchy Consistency. Whether the headings form a coherent outline, with a single root `<h1>` and no skipped levels. Let $h(i)$ be the level of the i -th of m headings in document order, and $c(h1)$ the number of `<h1>` elements.

$$I_{SI-2} = \alpha \cdot \mathbf{1}(c(h_1) = 1) + (1 - \alpha) \left(1 - \frac{\sum_{i=1}^{m-1} \max(0, h^{(i+1)} - h^{(i)} - 1)}{m - 1} \right)$$

We set $\alpha = 0.5$ as an equal-contribution design prior. When $m \leq 1$ the second term is taken as 1, and the result is clipped to $[0, 1]$.



SI-3, Semantic Element Fidelity. The share of interactive controls built with native semantic tags (<button>, <a>, <input>, <select>, <textarea>) instead of a generic <div> or with a click handler attached.

$$I_{SI-3} = \frac{|E_{\text{native}}|}{|E_{\text{native}}| + |E_{\text{generic}}|}$$

SI-4, Content-to-DOM Structural Ratio. A proxy for verbosity: the share of all DOM elements that are content-bearing tags (p, h1 to h6, article, section, li, td, th, blockquote, code).

$$I_{SI-4} = \frac{|V_{\text{content}}|}{|V_{\text{DOM}}|}$$

This is a normalized ratio, not an absolute measure of complexity. Two pages with the same ratio can differ greatly in raw DOM size.

SI-5, Dynamic DOM Stability. Under full dynamic execution, the structural stability of the set of elements E over a window Δt :

$$I_{SI-5} = 1 - \frac{|E_{t_0} \Delta E_{t_1}|}{|E_{t_0} \cup E_{t_1}|}$$

A static auditing tool cannot run client-side hydration, so it uses a static proxy, I_{SI-5} , which checks for the absence of inline script-driven document rewrites and anti-scraping obfuscation wrappers. Any report that uses the proxy should say so; it should not be presented as the dynamic measure.

$$SI = \frac{1}{5} \sum_{k=1}^5 I_{SI-k}$$

B. Pillar 2: Semantic Richness (SR)

SR-1, Domain Entity Structured Data Coverage. How many of the Schema.org entity classes expected for the page's archetype (for example Product for commerce or TechArticle for documentation) are declared in valid JSON-LD blocks.

$$I_{SR-1} = \frac{|E_{\text{declared}} \cap E_{\text{domain}}|}{|E_{\text{domain}}|}$$

SR-2, Structured Data Conformance. The share of declared JSON-LD blocks that are syntactically and schematically valid.

$$I_{SR-2} = \frac{|\text{valid JSON-LD blocks}|}{\max(1, |\text{declared JSON-LD blocks}|)}$$

SR-3, Non-Text Content Informational Value. The share of informational images (the set M_{info}) that carry a non-empty, descriptive alt attribute (the set M_{valid}).

$$I_{SR-3} = \frac{|M_{\text{valid}}|}{\max(1, |M_{\text{info}}|)}$$

SR-4, Programmatic Accessible Name Density. The share of interactive elements that have a programmatic accessible name through native text, aria-label or aria-labelledby.

$$I_{SR-4} = \frac{|\{e \in E_{\text{interactive}} : \text{AccName}(e) \neq \emptyset\}|}{|E_{\text{interactive}}|}$$

SR-5, Page Context and Entity Metadata Completeness. The presence of document-level metadata from the set $K = \{\text{og:title}, \text{og:description}, \text{og:image}, \text{og:url}, \text{canonical}\}$. An agent can read these tags as a quick description of the page before it walks the DOM.

$$I_{SR-5} = \frac{1}{|K|} \sum_{k \in K} \mathbf{1}(\text{tag } k \text{ present and non-empty})$$



$$SR = \frac{1}{5} \sum_{k=1}^5 I_{SR-k}$$

C. Pillar 3: Action Executability (AE)

AE-1, Keyboard and Programmatic Focusability. The share of interactive controls that can be reached through sequential focus order, without coordinate-based emulation.

$$I_{AE-1} = \frac{|\{e \in E_{\text{interactive}} : \text{focusable}(e) \wedge \text{tabIndex}(e) \geq 0\}|}{|E_{\text{interactive}}|}$$

AE-2, Action Target Contextual Disambiguation. A control is ambiguous when its accessible name matches that of another control in the same landmark region and nothing in its programmatic context distinguishes the two. Each ambiguous control lowers the score directly.

$$I_{AE-2} = 1 - \frac{|E_{\text{ambiguous}}|}{|E_{\text{interactive}}|}$$

AE-3, Form Control Association Integrity. The share of visible form controls F that are programmatically linked to an explicit label through `<label for="id">`, an enclosing `<label>`, or `aria-labelledby`.

$$I_{AE-3} = \frac{|\{f \in F : \text{hasLabel}(f)\}|}{|F|}$$

AE-4, Link Target Determinism. The share of hyperlinks whose target is a valid, resolvable URI, as opposed to an empty value, a bare #, or a script pseudo-protocol such as `javascript:void(0)`.

$$I_{AE-4} = \frac{|\{a \in L : \text{href}(a) \notin \{\emptyset, \#, \dots\}\}|}{|L|}$$

AE-5, Frictionless Programmatic Navigation. Whether an agent can reach the site's core flows without a blocking challenge. The indicator scores 1.0 when access is unrestricted, 0.5 when a soft warning appears, and 0.0 when a blocking challenge or firewall intercept stops the agent.

$$AE = \frac{1}{5} \sum_{k=1}^5 I_{AE-k}$$

D. Pillar 4: Machine Discoverability and Protocol Support (MDP)

This pillar has three parts: access policy (MDP-1), discovery (MDP-2 to MDP-4) and transport hygiene (MDP-5).

MDP-1, Declarative Crawling and Agent Directives. Whether the site publishes a `robots.txt` file and whether it contains rules for specific agent classes, such as the user-agent tokens that AI crawlers announce [26].

$$I_{MDP-1} = 0.5 \cdot \mathbf{1}(\text{robots.txt exists}) + 0.5 \cdot \mathbf{1}(\text{agent-class directives present})$$

MDP-2, Machine-Readable Sitemap Completeness. Scores 1.0 if a valid `sitemap.xml` is referenced in `robots.txt`, 0.5 if a sitemap exists but is not referenced, and 0.0 if none is found.

MDP-3, Agent Manifest Availability. Scores 1.0 if a well-formed `/lms.txt` file is present and 0.0 otherwise. `/lms.txt` is a community proposal [22], so this framework version treats it as experimental. Its absence costs only this one indicator.

MDP-4, Programmatic API Discovery. Scores 1.0 if the site advertises an OpenAPI specification or an MCP server manifest [21], 0.5 if it declares a REST endpoint without a machine-readable specification, and 0.0 if it advertises neither.

MDP-5, Transport Header Caching and Character Hygiene. Whether responses declare their character encoding and a valid caching policy.

$$I_{MDP-5} = 0.5 \cdot \mathbf{1}(\text{explicit UTF-8 charset}) + 0.5 \cdot \mathbf{1}(\text{valid Cache-Control})$$

$$MDP = \frac{1}{5} \sum_{k=1}^5 I_{MDP-k}$$



E. Pillar 5: Resilience and Error Recoverability (RR)

RR-1, HTTP Status Code Semantics. Scores 1 when error states return an appropriate 4xx or 5xx status and 0 when they return HTTP 200 with the error shown only as visible text. A static audit can check this only for the primary URL's response. Testing generated error states, such as an invalid form submission, needs a dynamic evaluation mode.

RR-2, Programmatic Error Surfacing. The share of validation and application error messages that are bound to their originating control through aria-describedby or announced through a live region (role="alert").

RR-3, Declared Dynamic-State Notification Support. In a static audit, written I_{RR-3} , this is the share of dynamic regions and dialogs that declare an ARIA live region (aria-live="polite", role="status") or a focus-management binding. A dynamic audit instead records whether live regions actually fire during asynchronous updates.

RR-4, Session and Timeout Transparency. How the application warns before a session expires:

$$I_{RR-4} = \begin{cases} 1.0 & \text{programmatic warning at least } T_w \text{ in advance} \\ 0.5 & \text{visual-only warning} \\ 0.0 & \text{session ends without notice} \end{cases}$$

T_w is a configurable warning threshold. Any value used should be reported as a heuristic prior until it has been calibrated against measured agent re-authentication latency.

RR-5, Resource Cache Revalidation Support. Whether responses carry the validators needed for conditional requests [25].

$$I_{RR-5} = 0.5 \cdot \mathbf{1}(\text{ETag present}) + 0.5 \cdot \mathbf{1}(\text{Last-Modified present})$$

$$RR = \frac{1}{5} \sum_{k=1}^5 I_{RR-k}$$

F. Composite Scores

The unweighted baseline treats all pillars equally and involves no subjective weighting:

$$AAS_{\text{base}} = \frac{1}{5} (SI + SR + AE + MDP + RR)$$

Pillar failures are not equally severe, however. A control the agent cannot click (Action Executability) can end a task outright, while a missing /lms.txt (Machine Discoverability) adds exploration overhead but does not by itself prevent completion. The failure-criticality-weighted score reflects this:

$$AAS_{\text{crit}} = 0.25 SI + 0.20 SR + 0.30 AE + 0.10 MDP + 0.15 RR$$

These weights are initial priors based on the severity argument above. They were not fitted to outcome data, because no such data exists yet for this framework. The same applies to the sub-coefficients inside indicators ($\alpha = 0.5$ in SI-2 and the 0.5/0.5 splits in MDP-1, MDP-5 and RR-5), which are equal-contribution design priors. Anyone using the scores should report them as such.

Table II gives provisional tiers for reading a composite score. The cut points are round numbers chosen for readability, not validated thresholds. Recalibrating them against real score distributions is one of the tasks for the empirical study in Section VIII.

TABLE II Provisional Interpretation Tiers for Composite Scores

Score range	Tier	Typical characteristics
0.80 to 1.00	Agent-Optimized	Complete semantic markup, structured JSON-LD, explicit affordances, machine manifests
0.60 to 0.79	Agent-Friendly	Well-structured native HTML and clear affordances; some gaps in structured data or manifests
0.40 to 0.59	Agent-Compatible	Conventional presentational site; little structured metadata, so agents rely on heuristic text parsing
0.20 to 0.39	Agent-Challenged	Deeply nested non-semantic containers, unlabeled controls, missing landmarks
0.00 to 0.19	Agent-Hostile	Aggressive interstitials, dynamic obfuscation, DOM that cannot be navigated



V. DESIGN GUIDELINES

Each guideline below follows from the indicator with the same number in Section IV.

A. Structural Interpretability

G-SI-1. Put every visible interface region inside a standard HTML5 landmark, with exactly one `<main>` per document. *G-SI-2.* Use a single `<h1>` and step down through heading levels without skipping any. *G-SI-3.* Use `<button>` for commands that change state and `<a>` for navigation; do not attach click handlers to `<div>` or `<span>`. *G-SI-4.* Remove wrapper elements that carry no content or semantics, so that content-bearing tags make up a larger share of the DOM and serialized pages cost fewer tokens. *G-SI-5.* Render critical content and controls in the initial HTML payload through server-side rendering or static generation, so that they do not shift after hydration.

B. Semantic Richness

G-SR-1. Embed complete, valid JSON-LD for the main Schema.org entities on every content and transactional page. *G-SR-2.* Validate JSON-LD against the Schema.org vocabulary before deployment and fill in the required properties. *G-SR-3.* Give informational images short, descriptive alt text and mark decorative images with `alt=""`. *G-SR-4.* Give every interactive control a non-empty accessible name, and label icon-only buttons with `aria-label`. *G-SR-5.* Declare the canonical URL, a description and OpenGraph tags in `<head>`.

C. Action Executability

G-AE-1. Prefer native focusable elements and use `tabindex="0"` only on custom elements that need it. *G-AE-2.* Keep accessible names unique within a landmark, and qualify repeated actions (for example `aria-label="Edit Order 1042"`). *G-AE-3.* Bind every form control to an explicit `<label>`. *G-AE-4.* Point navigation elements at resolvable URIs, not `javascript:void(0)` or a bare `#`. *G-AE-5.* Offer documented machine authorization, such as OAuth or API keys, for legitimate automated workflows. Bot defenses exist for good reasons, since LLM agents can also be used for attacks [27], but where a non-visual verification path works equally well, prefer it to a visual challenge.

D. Machine Discoverability and Protocol Support

G-MDP-1. Keep `/robots.txt` current, with explicit rules for agent classes. *G-MDP-2.* Publish an XML sitemap and reference it from `robots.txt`. *G-MDP-3.* Consider publishing `/llms.txt` as a low-cost discovery aid, keeping in mind that it is an emerging convention and not a ratified standard. *G-MDP-4.* Expose machine-readable API specifications (OpenAPI or MCP manifests) through `<link rel="service-doc">`. *G-MDP-5.* Always send an explicit Content-Type: `text/html; charset=UTF-8` header and a correct Cache-Control header.

E. Resilience and Error Recoverability

G-RR-1. Do not return HTTP 200 for an application error; use the appropriate 4xx or 5xx code. *G-RR-2.* Announce validation and server errors through `aria-live` or `role="alert"`, and bind them to the originating input with `aria-describedby`. *G-RR-3.* Move focus or trigger a live-region announcement when a dialog opens, content refreshes or a multi-step form advances. *G-RR-4.* Warn programmatically before a session expires, with a lead time based on the measured re-authentication latency of the agents you expect, not on a guess. *G-RR-5.* Send ETag and Last-Modified headers on transactional endpoints so that clients can make conditional requests.

VI. DISCUSSION

A. Designing for Agents and for People

We claim that building for agent accessibility and building for human accessibility support each other. Landmark regions, a coherent heading outline, explicit form labels and keyboard focus satisfy core WCAG 2.2 success criteria, including SC 1.3.1 Info and Relationships, SC 2.1.1 Keyboard and SC 2.4.6 Headings and Labels [6]. The same markup produces the clear accessibility tree an LLM agent needs in order to plan and act. The argument rests on a shared mechanism, since both consumers read the same tree. We have not tested it against paired human and agent task outcomes, and that test is future work. The overlap may also matter for regulation: accessibility obligations such as the European Accessibility Act [28] already push organizations toward the markup that AAMF rewards.



B. What the Framework Predicts

The indicator definitions suggest an asymmetry that we state as a falsifiable hypothesis, not a finding. Native HTML controls, anchors and form elements are the default output of most modern web frameworks. We therefore expect Action Executability to score highest across a broad, unselected sample of production sites, because frameworks default to semantic primitives for reasons unrelated to agents. Schema.org coverage (SR) and machine manifests such as /lms.txt (MDP), on the other hand, take extra engineering work that brings no immediate return to a site that is not courting agent traffic. We expect these two pillars to be the weakest in such a sample. The empirical study described in Section VIII is designed to test this.

VII. LIMITATIONS AND THE REQUIRED EMPIRICAL STUDY

The main limitation of this paper is also a scoping decision: it contains no empirical measurement. The prediction in Section VII-B comes from the design of the framework and remains untested. We describe here what a proper test would involve, so that the framework can be checked and, if necessary, refuted.

Sampling. An audit needs a pre-registered sampling frame, stratified by site archetype (for example documentation, developer platforms, e-commerce, news and forums, and knowledge bases), with enough sites per stratum to support the confidence intervals it reports. A handful of well-known sites chosen for convenience should not be analyzed with statistics that assume a random sample.

Dynamic execution. A static parser in the style of Cheerio or BeautifulSoup cannot evaluate SI-5, the dynamic error paths of RR-1, or RR-3 as originally defined. Measuring these indicators needs a scanner instrumented with Playwright or Puppeteer that runs real client-side hydration, instead of relying on static proxies.

Indicator validity. The 25 indicators and their sub-coefficients are design priors and have not been validated. An expert panel should assess their content validity. Later, Agent Accessibility Score (AAS) should be regressed on real agent task outcomes before the failure-criticality weights are treated as more than a starting hypothesis.

Link to agent performance. Whether AAS predicts agent task success, token cost or retry rate is an open question. Answering it means running real agents against real or controlled synthetic pages, over a task suite large enough to support statistical claims.

Reproducibility. Any audit should release its raw per-indicator output, the auditor's source code and the exact tool versions, so that every reported number can be re-derived independently.

We chose to publish the framework on its own, with these gaps stated openly, instead of attaching an audit that could not meet these standards.

VIII. CONCLUSION AND FUTURE WORK

This paper introduced the Agent Accessibility Measurement Framework. It defines agent accessibility within ISO/IEC 25010:2023, describes the five-layer Agent Interaction Stack, and specifies 25 indicators across five pillars, two composite scores and 25 engineering guidelines derived from the indicator definitions. We have made no empirical claims about production websites, because we do not yet have audit data collected to a standard we can defend.

Five pieces of work come next. The first is an open-source AAMF auditor instrumented with Playwright that supports full dynamic execution. The second is a pre-registered, adequately powered audit of a stratified sample of production sites, reporting indicator distributions with properly scoped confidence intervals. The third is an expert-panel review of the content validity of the indicators and their sub-coefficients. The fourth is an agent task study large enough to relate AAS to task success, token cost and retry rate. The fifth is a proposal, through a relevant W3C community group, for an agent-oriented extension to WCAG. This paper is the first step in that sequence and does not replace any of the later ones.



REFERENCES

1. T. B. Brown, B. Mann, N. Ryder, et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 1877-1901.
2. S. Zhou, F. F. Xu, H. Zhu, et al., "WebArena: A realistic web environment for building autonomous agents," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024, arXiv:2307.13854.
3. X. Deng, Y. Gu, B. Zheng, et al., "Mind2Web: Towards a generalist agent for the web," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023, arXiv:2306.06070.
4. McKinsey Global Institute, "The economic potential of generative AI: The next productivity frontier," McKinsey & Company, Jun. 2023.
5. H. Lai, X. Liu, I. L. Iong, S. Yao, Y. Chen, P. Shen, H. Yu, H. Zhang, X. Zhang, Y. Dong, and J. Tang, "AutoWebGLM: A large language model-based web navigating agent," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, 2024, arXiv:2404.03648.
6. W3C Web Accessibility Initiative, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, Oct. 2023.
7. *Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): Product Quality Model*, ISO/IEC 25010:2023, International Organization for Standardization, Geneva, Switzerland, 2023.
8. R. Nakano, J. Hilton, S. Balaji, et al., "WebGPT: Browser-assisted question-answering with human feedback," arXiv:2112.09332, 2021.
9. Gur, H. Furuta, A. Huang, et al., "A real-world WebAgent with planning, long context understanding, and program synthesis," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024, arXiv:2307.12856.
10. S. Yao, J. Zhao, D. Yu, et al., "ReAct: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023, arXiv:2210.03629.
11. D. Nguyen, J. Chen, Y. Wang, et al., "GUI agents: A survey," arXiv:2412.13501, 2024.
12. J. Y. Koh, R. Lo, L. Jang, et al., "VisualWebArena: Evaluating multimodal agents on realistic visual web tasks," in *Proc. 62nd Annu. Meeting of the Association for Computational Linguistics (ACL)*, 2024, arXiv:2401.13649.
13. Microsoft Playwright Team, "Accessibility testing," Playwright Documentation, 2024. [Online]. Available: <https://playwright.dev/docs/accessibility-testing>
14. W3C, "Accessible Rich Internet Applications (WAI-ARIA) 1.2," W3C Recommendation, Jun. 2023.
15. Levy, B. Wiesel, S. Marreed, A. Oved, A. Yaeli, and S. Shlomov, "ST-WebAgentBench: A benchmark for evaluating safety and trustworthiness in web agents," arXiv:2410.06703, 2024.
16. S. Zhong, K. Shen, and C. Xiong, "AgentWebBench: Benchmarking multi-agent coordination in agentic web," arXiv:2604.10938, Apr. 2026.
17. J. P. Bigham, C. M. Prince, and R. E. Ladner, "WebAnywhere: A screen reader on-the-go," in *Proc. Int. Cross-Disciplinary Conf. Web Accessibility (W4A)*, Beijing, China, 2008, pp. 73-82.
18. T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34-43, May 2001.
19. R. V. Guha, D. Brickley, and S. Macbeth, "Schema.org: Evolution of structured data on the web," *Communications of the ACM*, vol. 59, no. 2, pp. 44-51, Feb. 2016.
20. G. Kellogg, P.-A. Champin, and D. Longley, Eds., "JSON-LD 1.1: A JSON-based serialization for linked data," W3C Recommendation, Jul. 2020.
21. Anthropic, "Introducing the Model Context Protocol," Anthropic News, Nov. 2024. [Online]. Available: <https://www.anthropic.com/news/model-context-protocol>
22. J. Howard, "The /llms.txt file," proposal, Sep. 2024. [Online]. Available: <https://llmstxt.org>



23. Google, "Introduction to structured data markup in Google Search," Google Search Central Documentation, 2024.
24. L. Olsina and G. Rossi, "Measuring Web application quality with WebQEM," *IEEE MultiMedia*, vol. 9, no. 4, pp. 20-29, Oct.-Dec. 2002.
25. R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. California, Irvine, CA, USA, 2000.
26. OpenAI, "GPTBot," OpenAI Platform Documentation, 2023. [Online]. Available: <https://platform.openai.com/docs/gptbot>
27. R. Fang, R. Bindu, A. Gupta, and D. Kang, "LLM agents can autonomously exploit one-day vulnerabilities," arXiv:2404.08144, 2024.
28. European Parliament and Council of the European Union, "Directive (EU) 2019/882 on the accessibility requirements for products and services," *Official Journal of the European Union*, L 151, Jun. 2019.

