Grounding Persona-Driven Usability Simulation in Established Standards

Oussama Touir[1,*,†], Farah Barika Ktata[2,†] and Makram Soui[3,†]

[1]ISSATSO, University Of SOUSSE
[2]MIRACL LR05ES13, ISSATSO, University Of SOUSSE
[3]University of Michigan-Flint, United States

Abstract

As user interfaces (UIs) in safety-critical domains like automotive infotainment become increasingly complex, traditional usability testing faces a scalability bottleneck. While Large Language Models (LLMs) enable synthetic users that interact with interfaces and generate traces, these agents can be difficult to trust: they may fail to link simulated frustrations to diagnosable standards, and they can confound genuine usability defects with hallucinated constraints. We introduce a Simulation-Based Usability Testing Module (SUTM), a persona-driven, multi-agent workflow in which User-Agents execute Perceive–Decide–Act loops on rendered prototypes while a Supervisor-Agent verifies trace integrity and maps evidenced failures to established standards (ISO 9241-110 and Nielsen's heuristics). In an evaluation over $N = 45$ UI prototypes with expert ground truth and a single-agent baseline, verification increases diagnostic precision (ISO: 0.75→0.91) with only minor changes in recall, and reduces persona inconsistency from 45% to 0.8%. In a blind artifact review, developers report higher trust (median 5 vs. 4 on a 1–7 scale) and higher acceptance of recommendations (65% vs. 45%). We position SUTM as a responsible first-pass filter that augments expert evaluation through constrained persona governance, provenance bundles, and uncertainty-aware escalation.

Keywords

Usability Testing, AI Personas, Large Language Models, Generative Agents, Automated Evaluation, Responsible AI

1. Introduction and Motivation

Usability evaluation is a critical bottleneck in modern software development, particularly in high-stakes environments such as In-Vehicle Infotainment (IVI) systems. In these contexts, interaction errors—such as hard-to-read fonts or confusing navigation flows—can increase driver cognitive load. Traditional evaluation methods, such as lab-based user testing and expert heuristic reviews, remain the gold standard but are resource-intensive and difficult to scale across rapid iterative design cycles.

To address this scalability gap, recent research has pivoted toward automated evaluation. Early approaches relied on static analysis, using computer vision to predict tappability from pixels[1] or Vision-Language Models (VLMs) to critique static mockups[2]. However, static methods analyze screenshots in isolation and cannot detect temporal issues like broken navigation flows, state-dependent errors, or dead-ends. Behavioral approaches employ LLM-driven agents to simulate user interaction, as in systems like SimUser[3] and UXAgent[4]. While these systems introduce necessary dynamism, they can suffer from grounding failures: synthetic agents may report frustration or fail a task, but struggle to (i) diagnose the root cause, (ii) distinguish a structural defect from a hallucinated constraint, and (iii) link issues to established HCI standards.

We present the Simulation-Based Usability Testing Module (SUTM), designed to bridge behavioral simulation and standards-grounded reporting. SUTM operates within a multi-agent architecture: persona-based User-Agents generate interaction evidence (action traces, error logs, think-aloud self-

ACM CHI'26 Workshop on Responsible Use of AI Personas in Human-Centered Design and Research, April 13, 2026, Barcelona, Spain.
*Corresponding author.
†These authors contributed equally.
Touir.oussema@gmail.com (O. Touir); Farah.Barika.Ktata@issatso.u-sousse.tn (F. B. Ktata); msoui@umich.edu (M. Soui)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-08-28

CEUR-WS.org/Vol-4249/paper4.pdf

reports), while a Supervisor-Agent verifies trace integrity, aggregates recurrent failure modes, and maps evidenced issues to ISO 9241-110 and Nielsen's heuristics.

1.1. Design Goals

We derive five design goals for standards-grounded persona simulation:

- **Evidence-linked findings:** Report issues with concrete evidence anchors (trace steps and UI targets), not generic critique.
- **Grounded personas:** Enforce persona constraints across actions to reduce implicit-vs-explicit mismatches.
- **Standards-grounded diagnosis:** Link failures to established principles (e.g., ISO 9241-110, Nielsen) to support actionable remediation.
- **Verify-before-reporting:** Re-run simulations after proposed changes to confirm resolution and detect regressions.
- **Responsible governance:** Constrain persona generation, package provenance, and escalate uncertain cases for human review.

Beyond performance, we align with responsible AI-persona use. Prior work on human–AI collaboration and multi-agent orchestration highlights risks around opaque decisions, over-trust, and role ambiguity in agentic systems[5, 6]. We therefore frame SUTM as decision support with explicit governance: constrained persona generation, auditable traces, verification-before-reporting, and human escalation for uncertain recommendations.

1.2. Contributions

Our contributions are threefold: (1) a persona-grounded simulation loop for interaction-level usability testing, (2) a standards-based grounding and verification mechanism that improves recommendation traceability, and (3) a responsible deployment framework that positions synthetic users as a first-pass triage filter to augment (not replace) expert evaluation.

2. Related Work

We position SUTM at the intersection of automated usability evaluation, simulated user agents, and trust/verification in AI-assisted workflows. Across these threads, a recurring tension is *scale vs. trust*: automated systems can surface many candidate issues quickly, but practitioners need evidence, reproducibility, and governance to act on the output without over-trusting it[7, 5].

2.1. Automated and Static Usability Evaluation

Automated usability evaluation spans static checking, metric-based prediction, and data-driven critique. Pixel- and saliency-based models can predict tappability and other interaction cues[1], while VLM-driven critique can produce richer natural language feedback from screenshots or mockups[2]. Complementary work evaluates UI quality via engineered metrics and optimization (e.g., mobile GUI quality and aesthetics)[8, 9] or via learning-based scoring of UI designs[10]. Dataset-driven critique approaches also aim to standardize what "good" or "problematic" design looks like across large UI collections[11].

These approaches improve coverage for many observable issues, but they are inherently limited in capturing interaction-dependent breakdowns in multi-step flows (e.g., dead-ends after state changes, repeated backtracking, or failures that only manifest after form validation). They also often underspecify *what exactly in the UI caused the problem* and *how to reproduce it*, which can reduce actionability for developers.

2.2. Simulated Users and Agentic Usability Testing

LLM-driven agents have been proposed as scalable proxies for exploring UI behavior by interacting directly with interfaces[3, 4, 12]. This paradigm is appealing because interaction traces capture temporal friction (e.g., repeated backtracking, dead-ends, and state-dependent failures) that static critique may miss. Beyond web and mockup settings, multimodal agents have also been explored as smartphone users that navigate app screens to complete tasks[13].

However, simulated-user outputs are often difficult to aggregate, validate, and compare. Without explicit evidence anchors and consistent task definitions, it can be unclear whether an agent's narrative reflects a genuine usability barrier or a model artifact. In practice, usable feedback requires not only that the agent *acts*, but that the system records reproducible traces and ties failures to concrete UI targets that developers can inspect.

2.3. Grounding, Verification, and Responsible Use

A central challenge is calibrated trust. Systematic reviews of AI-assisted UX evaluation emphasize risks around hallucinated outputs, contextual misinterpretation, and bias, motivating human-in-the-loop and explainable workflows[7]. In multi-agent systems, role ambiguity and opaque coordination can further amplify over-trust and make responsibility unclear[6]. Prior work on human–AI collaborative UX analysis similarly argues for assistive tools that keep evaluators in the loop rather than replacing judgment[5].

SUTM operationalizes these concerns for persona-driven simulation by separating acting User-Agents from a supervising verifier, enforcing constrained persona schemas (to reduce unsupported stereotyping), and packaging provenance artifacts. We also draw on responsible accessibility tooling and remediation work that emphasizes verification and human oversight[14, 15].

3. System Implementation

SUTM simulates persona-driven interaction on UI prototypes and produces actionable, standards-backed feedback. Figure 1 illustrates the architecture.

3.1. Inputs and Outputs

Inputs are (i) a rendered UI prototype and (ii) a short purpose/context description (task goal and intended user). **Outputs** include persona profiles, action traces, error logs, think-aloud self-reports, standards-mapped issue reports with evidence anchors, and verification outcomes.

3.2. Orchestration, Typed Schemas, and Privacy

SUTM is implemented as a pipeline that orchestrates role-specialized agents with headless browser automation. A single run carries forward structured state: UI context, persona profiles, simulation traces, normalized issues, patch proposals, unified patches, and verification outcomes.

To reduce brittleness, all inter-agent messages are constrained by typed schemas (e.g., `PersonaProfile`, `IssueReport`, `PatchProposal`, `DiagnosticReport`) and validated before they are consumed by downstream stages. This schema boundary serves as an additional guardrail against malformed outputs and supports consistent logging.

For each evaluated UI, SUTM writes a structured diagnostic report (machine-readable JSON plus a human-readable summary) and, when patching is enabled, a patched artifact alongside a record of which patches applied successfully. The system runs locally except for LLM API calls; to reduce exposure of sensitive UI content, deployments can truncate or redact long HTML/DOM payloads while preserving the evidence anchors needed for reproducibility.

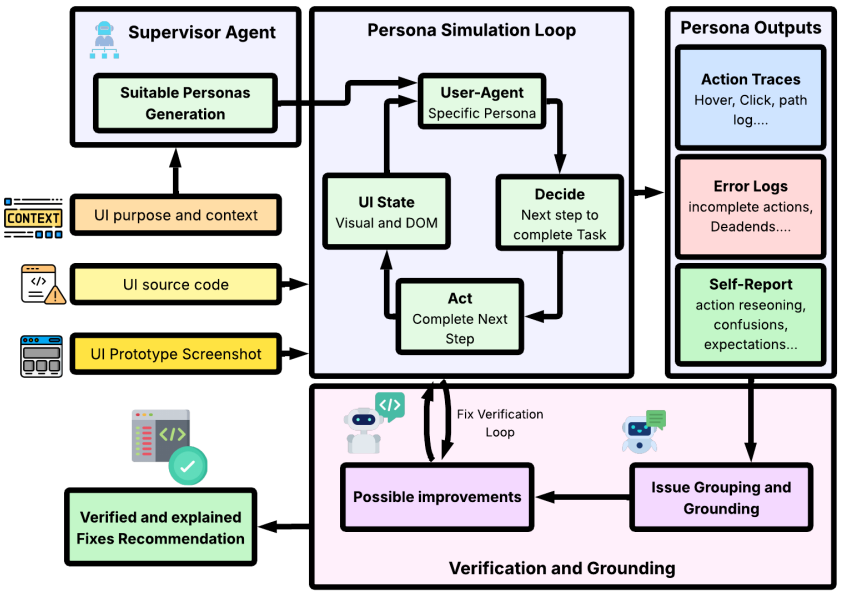

Figure 1: Proposed grounding persona-driven simulation framework.

3.3. Supervisor-Agent and Persona Generation

The process begins with a Supervisor-Agent that synthesizes high-fidelity personas from the UI purpose/context and source structure. Personas are parameterized by task goals, technical literacy, and accessibility constraints to probe edge cases that generic testing may miss. In practice, the Supervisor generates a small *persona pack* where each persona differs along at least one constraint dimension (e.g., novice vs. power-user strategies, or accessibility-stress constraints).

For example, our evaluation includes profiles such as *Martha R.*, a low-tech-literacy persona who relies heavily on explicit visual cues, and *David L.*, a deuteranope persona whose goal is to filter products by color (a stress test for color-dependent controls).

3.3.1. Persona schema

Each persona follows a constrained schema to reduce demographic stereotyping and improve auditability:

- Persona ID/name and task goal
- Expertise/tech literacy level
- Accessibility constraints (visual, motor, cognitive) when applicable
- Interaction strategy hints (e.g., prefer explicit cues)
- Step budget / stopping criteria

3.4. Persona Simulation Loop (Perceive–Decide–Act)

Each User-Agent executes a Perceive–Decide–Act loop. The agent perceives the UI state via a multimodal snapshot (visual layout + interactive structure), decides the next action conditioned on persona constraints and task progress, and acts (click/scroll/type). User-Agents emit three data streams: Action Traces, Error Logs, and Subjective Self-Reports.

3.5. Trace Integrity Verification and Standards Grounding

Raw agent traces can be noisy. To reduce grounding failures, the Supervisor verifies trace integrity before promoting issues:

- **Constraint compliance:** actions and rationales remain consistent with persona constraints.
- **Evidence plausibility:** referenced UI targets exist at the cited step and can be anchored reliably.
- **Trace coherence:** the agent pursues the task goal without unexplained leaps or contradictions.

When verification fails, the system downgrades the issue or selectively re-simulates the unstable segment under tightened constraints. Verified failures are mapped to relevant principles in ISO 9241-110 and Nielsen's heuristics, producing a standards-grounded, evidence-linked report.

3.6. Issue Normalization and Grouping

To make simulated-user output usable at scale, the Supervisor normalizes each candidate issue into a common schema: issue type, evidence anchor (trace step + UI target), mapped guideline, free-text explanation, and confidence. Normalization enables aggregation across personas and reduces the risk that slightly different natural-language critiques are treated as separate problems.

3.7. Recommendation Synthesis and Conflict Handling

For each failure-mode cluster, SUTM proposes the *smallest safe patch* that addresses the issue while minimizing unintended side effects. Patches can take three forms:

- **HTML attribute edits** (e.g., adding an accessible name via `aria-label` or improving form labeling),
- **CSS rule injection** (e.g., improving contrast or focus visibility), or
- **JavaScript snippets** when markup/CSS alone cannot repair the interaction breakdown.

Each patch proposal includes (i) a before-snippet anchor, (ii) an after-snippet change, (iii) a target selector/element, (iv) optional preconditions (e.g., only if an element lacks an accessible name), and (v) a rationale linking the evidence anchor to the relevant standard mapping.

When multiple clusters yield patches in the same UI, proposals can overlap. SUTM therefore detects conflicts when two patches (a) target the same selector/element, (b) modify the same attribute/property, or (c) introduce colliding CSS rules (specificity/order). Conflicts are resolved by rewriting proposals into a unified patch set or downgrading risky changes for human review.

3.8. Patch Application and Deterministic Verification

Patch application is treated as a first-class step because generated edits can be brittle if anchors drift across formatting, minification, or templating. SUTM applies patches using increasingly permissive matching strategies: (1) exact before-snippet match, (2) whitespace-normalized match, and (3) attribute-targeted matching when snippet context is unstable. All application outcomes are logged so developers can inspect which edits were applied cleanly vs. skipped.

After applying a unified patch set, SUTM re-runs the same persona pack and tasks. A recommendation is only promoted when (i) the previously failing behavior no longer triggers under the same persona/task and (ii) verification does not detect a high-severity regression in the exercised paths. The report records verified resolutions, unresolved issues, skipped patches, and any detected regressions.

3.9. Execution Guards and Reproducibility

To prevent degenerate simulations (e.g., repeated clicking, infinite backtracking, or prolonged observation without progress), each persona run enforces a step budget and lightweight guards such as repeated-action detection and consecutive no-op observation limits. For reproducibility and auditability, SUTM assigns stable IDs to personas and trace steps and records the artifacts needed for re-play (persona profile, trace, error logs, evidence anchors, and verification outcomes).

3.10. Verify-Before-Reporting

When SUTM proposes a concrete change (e.g., adding text labels to reduce ambiguity), it can apply the change to the prototype artifact and re-run the same personas on the modified UI. Recommendations are only promoted when the previously failing behavior no longer triggers under the same persona/task, and when no high-severity regressions are observed in the exercised paths.

3.11. Responsible Persona Governance

SUTM embeds governance controls at generation, simulation, and reporting time: constrained persona schemas, provenance bundles (persona prompt, traces, evidence anchors, mapped standards, verification outcomes), and uncertainty-aware escalation for weak-evidence recommendations.

4. Study Design

We evaluate SUTM along three axes: diagnostic accuracy vs. expert ground truth, persona grounding/fidelity, and developer-perceived utility.

4.1. Research Questions

- **RQ1 (Diagnostic accuracy):** How accurately does SUTM detect standards-mapped usability violations compared to experts, and does verification reduce false positives relative to a single-agent baseline?
- **RQ2 (Grounding and persona fidelity):** To what extent does SUTM reduce persona–behavior mismatches compared to a baseline without verification?
- **RQ3 (Developer utility):** How helpful, actionable, and trustworthy are SUTM artifacts for developers?

4.2. Dataset

We evaluate on $N = 45$ UI prototypes representing common interaction patterns (e.g., onboarding/login, checkout, booking, account recovery). We focus on prototypes that can be rendered and replayed deterministically so that interaction traces and verify-before-reporting re-simulations are reproducible.

These prototypes span a range of complexity and interaction modalities (e.g., scrolling, form entry, and multi-step navigation) to surface both surface-level and state-dependent breakdowns that are difficult to capture via static critique alone.

4.3. Study Conditions

We compare three conditions that isolate the effect of verification and supervision:

- **Full:** SUTM with persona packs, Perceive–Decide–Act simulation, trace integrity verification, issue grouping, patch synthesis with conflict handling, and verify-before-reporting re-simulation.
- **Base:** a single-agent baseline that produces issues and recommendations from the same UI and persona description, but without Supervisor verification, grouping, conflict handling, or verification re-simulation.
- **Expert:** expert standards/heuristics inspection producing labeled violations with evidence anchors.

To isolate architectural effects, we hold the underlying model family constant across Full and Base and keep the persona/task definitions aligned.

This isolates the contribution of supervision and verification (trace integrity checks, evidence-linked standards grounding, and verify-before-reporting re-simulation) rather than differences in task specification or persona coverage.

Table 1
Study conditions and enabled components.

Cond.	Enabled components
Full	Persona pack; interaction traces; trace integrity verification; standards mapping with evidence anchors; issue grouping; patch synthesis with conflict detection; verify-before-reporting re-simulation.
Base	Single agent produces issues and recommendations without Supervisor verification, grouping, or re-simulation.
Expert	Manual inspection using standards/heuristics with evidence anchors.

4.4. Tasks and Persona Packs

For each UI, the Supervisor generates a compact persona pack that varies along literacy and accessibility constraints. We include a small set of *stress personas* (e.g., strict accessibility constraints) to probe boundary conditions while keeping the pack small enough for iterative verification.

We structure the protocol around five tasks:

- **T01 Core workflow simulation:** attempt an end-to-end goal path and log action traces.
- **T02 Standards-mapped violation detection:** report issues with guideline IDs and evidence anchors.
- **T03 Persona fidelity stress test:** measure persona–behavior mismatch under strict constraints.
- **T04 Recommendation and verification loop:** generate patches, reconcile conflicts, apply changes, and verify via re-simulation (Full only).
- **T05 Developer blind artifact review:** collect developer ratings over blind-coded artifact packages.

For T01–T04, each persona run terminates when the task goal is achieved, the agent reaches a dead-end, or the step budget is exhausted. We log stable step IDs, UI targets, and error events so that each reported issue can be traced back to a concrete interaction segment; in the Full condition, the Supervisor may trigger selective re-simulation of unstable segments when evidence plausibility or constraint compliance is uncertain.

4.5. Expert Ground Truth Construction

For expert ground truth, we recruited 12 HCI experts. Each UI was labeled independently by three experts using a structured form (guideline ID, evidence anchor, severity, and confidence), grounded in ISO 9241-110 and Nielsen's heuristics (and accessibility criteria such as WCAG when applicable). We compute inter-rater reliability (Fleiss' κ)[16]; across the dataset we obtained $\kappa = 0.78$, indicating substantial agreement. We adjudicate only disputed items via a coordinator.

To support direct scoring against system outputs, experts were instructed to make evidence anchors localizable (specific UI target plus a minimal reproduction cue) so that labels can be aligned to trace-anchored reports.

4.6. Developer Artifact Review

Developers review blind-coded artifact packages and rate trust, workload, and utility. They also decide whether to accept, refine, or reject recommendations, providing rationale tags.

Each package includes the persona/task context and standards-mapped findings with evidence anchors (trace step + UI target), and when available, patch proposals with application and verification outcomes.

We recruited 12 developers (balanced across novice and intermediate experience levels). Packages are counterbalanced and condition-identifying metadata is removed by a coordinator. After each package,

developers complete trust ratings (1–7) inspired by trust-in-automation constructs[17], NASA-TLX workload items (0–100)[18], and utility/correctness rubrics (1–5), followed by an ACCEPT/REFINE/REJECT decision.

Counterbalancing and blinding reduce order and condition bias, supporting a fair comparison of Full vs. Base artifacts.

4.7. Metrics

We register outcomes aligned with the research questions:

For RQ1, we treat evidence anchors as the unit of comparison: a system report is considered a match when it refers to the same UI target/interaction segment and guideline family as an expert-labeled violation within the same UI. For RQ2, we operationalize persona inconsistency and hallucination as verifiable contradictions (e.g., explicit acknowledgment of a constraint or observed UI state followed by a conflicting action or claim); selective re-simulations / UI counts verifier-triggered re-runs used to resolve unstable evidence before reporting.

- **RQ1:** detection precision/recall/F1 relative to expert ground truth (reported for ISO 9241-110 overall and Nielsen overall).
- **RQ2:** persona inconsistency rate and hallucination rate, operationalized as an explicit constraint acknowledgment followed by a contradictory action/claim; we also report selective re-simulations per UI.
- **RQ3:** developer trust, workload, utility, fix correctness ratings, and ACCEPT/REFINE/REJECT rates.

5. Results

5.1. Diagnostic Accuracy vs. Experts

Table 2 reports detection performance against expert annotations.

Across both guideline families, the primary effect of supervision and verification is improved precision, consistent with fewer false positives when evidence plausibility and persona constraint compliance are checked. Recall changes are small and mixed across guideline families, suggesting that verification primarily filters unreliable or weakly grounded findings rather than reducing coverage.

Table 2
Detection performance against expert ground truth.

Category	Cond.	Prec.	Rec.	F1
ISO 9241-110 (overall)	Full	0.91	0.85	0.88
ISO 9241-110 (overall)	Base	0.75	0.88	0.81
Nielsen (overall)	Full	0.96	0.92	0.94
Nielsen (overall)	Base	0.78	0.90	0.84

5.2. Grounding and Persona Fidelity

Table 3 summarizes persona grounding metrics.

Table 3
Grounding and persona fidelity metrics.

Metric	Full	Base
Persona inconsistency rate	0.8%	45%
Hallucination rate	0.9%	45%
Selective re-simulations / UI	0.8	–

Persona inconsistency captures cases where an agent's actions or explanations contradict its stated persona constraints (e.g., acknowledging a limitation but then proceeding as if the limitation does not apply). Hallucination rate captures claims about UI state or failure causes that are not supported by the recorded trace and evidence anchors. Selective re-simulations / UI measures how often the Supervisor re-runs unstable segments under tightened checks to resolve evidence plausibility or constraint compliance before reporting.

5.3. Patch Synthesis and Verification Outcomes

Table 4 summarizes outcomes for patch synthesis, application, and verify-before-reporting re-simulation. Conflicts arise when independently generated fixes overlap (e.g., multiple changes to the same control or colliding CSS rules). The system resolves most overlaps before application, but a non-trivial fraction of patches can still fail to apply cleanly due to anchor drift.

Verification re-simulation grounds remediation in interaction evidence: a patch is counted as passing when the original issue no longer triggers under the same persona/task and no high-severity regressions are observed in the exercised paths.

We count a patch as *applied successfully* when its anchor can be located and the edit is inserted into the target artifact without conflicts; failed applications typically reflect anchor drift, missing selectors, or downgraded overlaps. Verification and regression outcomes are bounded by the exercised personas and tasks, so the reported regression rate reflects detected issues within the verified scope rather than full-coverage testing.

Table 4
Patch synthesis and verification outcomes (Full condition).

Outcome	Value
Conflicting proposals / run	0.8
Patches applied successfully	78%
Verification pass rate	85%
Regression rate	7%

5.4. Developer Utility

Table 5 reports developer ratings for blind-coded artifact packages.

Table 5
Developer ratings per artifact package. Values are reported as median (IQR).

Measure	Full	Base
Trust (1–7)	5 (1)	4 (2)
NASA-TLX (0–100)	35 (15)	50 (20)
Utility (1–5)	4 (1)	3 (1)
Fix correctness (1–5)	4 (1)	3 (1)
ACCEPT rate	65%	45%

We report medians and interquartile ranges (IQR) for ordinal ratings to summarize central tendency robustly across packages. ACCEPT indicates the developer would merge a recommendation as-is; REFINE indicates the recommendation is directionally correct but needs edits; REJECT indicates it is not suitable in its current form.

5.5. Qualitative Vignettes

Accessibility-driven ambiguity. In one scenario, *David L.* attempted to filter products by color but failed due to ambiguous color swatches. The trace anchored the failure to the swatch controls and

Table 6
Risk mitigation framework for responsible persona-driven usability simulation.

Risk	Example	Mitigation in SUTM	Residual Risk
Persona bias	Persona over-generalizes user behavior from stereotypes	Constrained persona schema; accessibility- and task-grounded attributes only; Supervisor review before execution	Subtle bias can persist in prompt templates
Hallucinated diagnosis	Agent reports a false cause for task failure	Verification-before-reporting; cross-persona agreement checks; standards-linked evidence required	Rare false positives under sparse traces
Over-trust by developers	Teams accept agent output without scrutiny	Human-in-the-loop sign-off for high-impact changes; uncertainty flags on weak evidence	Automation bias in time-constrained teams
Low traceability	Recommendation appears without clear rationale	Provenance bundle: prompt, action trace, failed step, guideline mapping, verification outcome	Audit overhead for large batches

the self-report documented confusion. The Supervisor grounded the issue in self-descriptiveness and visibility principles and proposed adding text labels; re-simulation confirmed the task completed.

Interaction-dependent dead-end. In a multi-step checkout flow, personas repeatedly navigated between steps without finding the next action because the primary button changed placement after validation. Static critique did not flag the issue, while interaction traces revealed backtracking and timeouts; standards mapping linked the failure to consistency and error tolerance.

6. Discussion

SUTM is designed to make persona-driven simulation more trustworthy and actionable by combining (i) persona-grounded interaction evidence, (ii) standards mapping, (iii) synthesis of patchable failure modes, and (iv) verify-before-reporting re-simulation. The results suggest that supervision and verification improve diagnostic precision, reduce grounding failures, and increase developer trust and acceptance.

6.1. Why Verification Matters for Simulated Users

Many agentic evaluation workflows rely on a single model to both act and explain. This can lead to overconfident post-hoc rationalization ("I failed, therefore the UI is bad") and to critiques that are difficult to reproduce. By separating acting personas from a verifying Supervisor, SUTM treats issue reports as hypotheses that require corroboration: constraint compliance, evidence plausibility, and trace coherence are checked before findings are promoted.

Verify-before-reporting further turns remediation into a testable claim. A recommendation is only reported as successful when the original failure no longer triggers under the same persona/task and verification does not detect a high-severity regression in the exercised paths. This structure supports calibrated developer trust by making it explicit what was verified, what was downgraded, and what remains uncertain.

6.2. Actionability Requires Synthesis, Not Just Generation

Developer-facing usability feedback must be localizable (what element? what evidence?) and implementable (what change?). Evidence anchors and standards mapping help triage, but the practical bottleneck is producing a minimal, mergeable patch. SUTM addresses this by (i) normalizing and

grouping redundant critiques into failure modes, (ii) proposing smallest-safe patches per cluster, and (iii) reconciling overlaps via conflict detection before application.

Engineering details also matter for actionability. Patch application can fail when anchors drift (formatting, templates, minification), so SUTM logs application outcomes and uses explicit matching fallbacks. Even with fallbacks, some patches remain unsafe or unverifiable; these cases are surfaced as candidates for human review rather than silently applied.

6.3. Boundaries of Persona-Driven Simulation

The quantitative results highlight a boundary condition: short-horizon simulations are better suited to interaction-level breakdowns than to global, subjective constructs such as long-term learnability. Similarly, verification is bounded by the specified tasks and personas; it can miss regressions or barriers outside the exercised paths. These limitations motivate reporting verification scope explicitly and treating simulation as a first-pass filter rather than a replacement for expert evaluation.

7. Responsible AI Considerations and Limitations

We position persona-driven simulation as a high-volume first-pass triage mechanism, not an autonomous final evaluator. Table 6 summarizes key risks, mitigations, and residual risks.

7.1. Interpreting the Risk-Mitigation Matrix

The risk-mitigation matrix is intended to support practical adoption decisions. It makes explicit that safeguards lower risk but do not eliminate it, and it clarifies when escalation to human experts is mandatory (e.g., weak evidence, unverifiable traces, or high-impact recommendations). In particular, pairing provenance bundles with verification outcomes helps teams distinguish reproducible interaction evidence from plausible-but-uncertain critiques.

7.2. Deployment Commitments

We make three deployment commitments. First, **human oversight**: high-impact recommendations remain subject to expert validation. Second, **transparency**: each recommendation is packaged with evidence anchors and standards mappings so teams can inspect why it was produced. Third, **bounded autonomy**: unresolved conflicts across personas or unverifiable traces are escalated rather than automatically converted into design actions.

7.3. Limitations

Scope of interfaces. Our evaluation focuses on UI prototypes that can be rendered and replayed deterministically (e.g., HTML/CSS/JS artifacts). Highly dynamic single-page applications, authentication-gated experiences, and flows that depend on server-side state can limit simulation fidelity. Native mobile and IVI applications may require additional instrumentation to expose stable evidence anchors and to support deterministic re-play.

Short-horizon evaluation. Some usability concepts (e.g., learnability over time, habit formation, sustained comprehension) are difficult to assess via short interaction traces. Similarly, some accessibility experiences are not well approximated by short simulations unless the system explicitly models assistive-technology workflows.

Verification coverage. Verification is bounded by the exercised personas and tasks. It can miss regressions or barriers outside the simulated paths and it may under-represent rare but high-impact edge cases. We therefore report verification scope explicitly and treat unresolved/weak-evidence issues as candidates for escalation rather than as final judgments.

Model and prompt dependence. Agent behavior can shift across model versions, prompt changes, rate-limiting artifacts, and UI rendering differences. Deployments should freeze configurations when possible, version prompts, and log deviations so that changes in results can be diagnosed.

Cost and scalability. Verification and selective re-simulation improve reliability but increase computation. SUTM mitigates this by re-running only unstable segments/personas when integrity checks fail, but teams should still expect a quality–cost trade-off when scaling to large UI surfaces.

Privacy and sensitive content. Because LLM calls may transmit UI text and structure, deployments should minimize the data sent to external providers (e.g., truncation/redaction) and ensure configurations match organizational policies. Provenance bundles should also be handled carefully when UIs contain sensitive user data.

8. Conclusion

SUTM grounds persona-driven usability simulation in established HCI standards using trace integrity verification and verify-before-reporting. Results suggest that verification and evidence anchoring improve diagnostic precision, reduce grounding failures, and increase developer trust, supporting a responsible workflow where simulated users augment (but do not replace) expert evaluation.

Declaration on Generative AI

During the preparation of this work, the author(s) used GPT-5 in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] E. Schoop, X. Zhou, G. Li, Z. Chen, B. Hartmann, Y. Li, Predicting and explaining mobile ui tappability with vision modeling and saliency analysis, Conference on Human Factors in Computing Systems - Proceedings (2022). doi:10.1145/3491102.3517497.

[2] P. Duan, J. Warner, Y. Li, B. Hartmann, Generating automatic feedback on ui mockups with large language models, in: Conference on Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, 2024. doi:10.1145/3613904.3642782.

[3] W. Xiang, H. Zhu, S. Lou, X. Chen, Z. Pan, Y. Jin, S. Chen, L. Sun, Simuser: Generating usability feedback by simulating various users interacting with mobile applications, in: Conference on Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, 2024. doi:10.1145/3613904.3642481.

[4] Y. Lu, B. Yao, H. Gu, J. Huang, Z. J. Wang, Y. Li, J. Gesi, Q. He, T. J. J. Li, D. Wang, Uxagent: An llm agent-based usability testing framework for web design, Conference on Human Factors in Computing Systems - Proceedings (2025). doi:10.1145/3706599.3719729.

[5] E. Kuang, Crafting human-ai collaborative analysis for user experience evaluation, in: Conference on Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, 2023. doi:10.1145/3544549.3577042.

[6] S. Schömbs, Y. Zhang, J. Goncalves, W. Johal, From conversation to orchestration: Hci challenges and opportunities in interactive multi-agentic systems, HAI 2025 - Proceedings of the 13th International Conference on Human-Agent Interaction (2025) 158–168. doi:10.1145/3765766.3765795.

[7] J. Liu, Ai in automated and remote ux evaluation: A systematic review (2014–2024), 2025. doi:10.1155/ahci/7442179.

[8] M. Soui, M. Chouchane, M. W. Mkaouer, M. Kessentini, K. Ghedira, Assessing the quality of mobile graphical user interfaces using multi-objective optimization, Soft Computing 24 (2020) 7685–7714. doi:10.1007/s00500-019-04391-8.

[9] N. Bessghaier, M. Soui, C. Kolski, M. Chouchane, On the detection of structural aesthetic defects of android mobile user interfaces with a metrics-based tool, ACM Transactions on Interactive Intelligent Systems 11 (2021). doi:10.1145/3410468.

[10] J. Wu, Y.-H. Peng, A. X. Y. Li, A. Swearngin, J. Bigham, J. Nichols, Uiclip: A data-driven model for assessing user interface design, in: Proceedings of the ACM Symposium on User Interface Software and Technology (UIST), 2024.

[11] P. Duan, C. Y. Cheng, G. Li, B. Hartmann, Y. Li, Uicrit: Enhancing automated design evaluation with a ui critique dataset, in: UIST 2024 - Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology, Association for Computing Machinery, Inc, 2024. doi:10.1145/3654777.3676381.

[12] Y. Lu, T.-Y. Hsu, H. Gu, L. Cui, Y. Xie, W. Headden, B. Yao, A. Veeragouni, J. Liu, S. Nag, J. Wang, D. Wang, Agenta/b: Automated and scalable web a/btesting with interactive llm agents (2026).

[13] C. Zhang, Z. Yang, J. Liu, Y. Li, Y. Han, X. Chen, Z. Huang, B. Fu, G. Yu, Appagent: Multimodal agents as smartphone users, in: Conference on Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, 2025. doi:10.1145/3706598.3713600.

[14] A. Othman, A. Dhouib, A. N. A. Jabor, Fostering websites accessibility: A case study on the use of the large language models chatgpt for automatic remediation, in: ACM International Conference Proceeding Series, Association for Computing Machinery, 2023, pp. 707–713. doi:10.1145/3594806.3596542.

[15] P. Mowar, Y. H. Peng, J. Wu, A. Steinfeld, J. P. Bigham, Codea11y: Making ai coding assistants useful for accessible web development, in: Conference on Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, 2025. doi:10.1145/3706598.3713335.

[16] J. L. Fleiss, Measuring nominal scale agreement among many raters, Psychological Bulletin 76 (1971) 378–382. doi:10.1037/H0031619.

[17] J.-Y. Jian, A. M. Bisantz, C. G. Drury, Foundations for an empirically determined scale of trust in automated systems, International Journal of Cognitive Ergonomics 4 (2000) 53–71. doi:10.1207/s15327566ijce0401_04.

[18] S. G. Hart, L. E. Staveland, Development of nasa-tlx (task load index): Results of empirical and theoretical research, Advances in Psychology 52 (1988) 139–183. doi:10.1016/S0166-4115(08)62386-9.