AUTOMMAS: Automating Test Case Generation from User Manuals via Multi-Agent Systems

Andrés D. Peralta
Federal University of Amazonas
Manaus, Brazil
andres@icomp.ufam.edu.br

Gabriel Carvalho
Federal University of Amazonas
Manaus, Brazil
gabriel.pacheco@icomp.ufam.edu.br

Igor Lima
Federal University of Amazonas
Manaus, Brazil
igor.lima@icomp.ufam.edu.br

Nilton S. Nascimento
Federal University of Amazonas
Manaus, Brazil
nilton.nascimento@icomp.ufam.edu.br

Yan Soares
Federal University of Amazonas
Manaus, Brazil
yan.soares@icomp.ufam.edu.br

Nikson Ferreira
INDT
Manaus, Brazil
nikson.ferreira@indt.org.br

Hallyson Melo
INDT
Manaus, Brazil
hallyson.melo@indt.org.br

Bruno Gadelha
Federal University of Amazonas
Manaus, Brazil
bruno@icomp.ufam.edu.br

André Carvalho
Federal University of Amazonas
Manaus, Brazil
andre@icomp.ufam.edu.br

Abstract

User Interface (UI) test automation is hampered by automation fragility, making reliable Test Case (TC) generation a persistent industrial challenge. User manuals are a rich, semi-structured source of end-to-end Use Cases (UCs) that existing approaches largely overlook. Large Language Models (LLMs) can mine this source, but single-agent pipelines suffer from context degradation and hallucinations. We propose AUTOMMAS, a multi-agent system that generates TCs from mobile device PDF user manuals in two phases: a multimodal agent extracts UCs page by page, after which an iterative loop generates one TC per UC while an independent judge agent corrects inconsistencies before acceptance. Compared with an Optical Character Recognition (OCR) plus LLM baseline and two commercial LLMs (Gemini 3.5 Flash and GPT-5.5 Instant), AUTOMMAS attains 87% extraction validity and 97% TC acceptance, while validating 3.6× faster than human review. These results provide initial evidence of the viability of agent-based architectures for automated test generation in continuous testing environments for mobile devices.

Keywords

Automated Validation, LLMs, Multi-Agent Systems, Prompt Engineering, Test Automation

1 Introduction

Continuous Integration and Continuous Deployment (CI/CD) pipelines in modern software engineering impose increasing demands on Quality Assurance (QA) processes [2]. UI test automation has historically been affected by so-called automation fragility, in which dynamic rendering or minor structural changes in the codebase cause widespread failures across the test suite even when the underlying business logic remains unchanged [4, 9]. Beyond this fragility, manually authoring TCs remains a labor-intensive task, motivating the search for reliable sources from which tests can be generated automatically. Mobile device user manuals represent one such source that remains largely unexplored: they document, through natural language and associated screenshots, the main functional procedures supported by a product, providing a natural foundation for UCs from which TCs can be derived and bridging the gap between high-level procedures and low-level test logic [14].

Recent advancements in LLMs offer a promising mechanism to bridge this gap by extracting logical flows directly from natural language descriptions [1]. However, adopting monolithic, single-agent LLMs for end-to-end test generation introduces significant limitations. Such architectures are susceptible to context degradation when processing long documents [11] and tend to produce hallucinations, generating syntactically valid yet functionally invalid interactions, such as referencing interface elements that do not exist in the device or assuming application states that the manual never describes [7]. In continuous testing environments, undetected hallucinations introduce systematic false positives, compromising the reliability of the QA pipeline.

This work targets two well-established software engineering artifacts. A *Use Case* is defined as a complete multi-step user procedure aimed at a single functional goal [6]. From each UC, the system generates a structured *Test Case* conforming to the five-component schema of Black Box test documentation (Title, Summary, Initial Setup, Steps, Expected Results), as standardized in IEEE 829 [5].

To address this challenge, we propose AUTOMMAS, a two-phase multi-agent architecture that automates TC generation from unstructured PDF user manuals. AUTOMMAS leverages Multimodal Large Language Models (MLLMs) to process both the textual content and the visual UI references embedded in such manuals: a multimodal extraction agent distills documented procedures into structured UCs, and an iterative loop generates one TC per UC while an independent judge agent verifies it against structural and semantic criteria before acceptance. Beyond the architecture itself, this work makes two methodological contributions: we separate generation from validation through an independent iterative loop that reduces the self-confirmation bias of single-agent approaches, and we define a blinded QA-expert protocol to assess extraction validity and TC quality in the absence of a formal ground truth.

This work addresses two central research questions. **RQ1** How can a multimodal multi-agent framework be designed to automatically extract UCs and generate structured, manually executable UI TCs from visually rich and unstructured mobile device PDF user manuals? **RQ2** How effective is an independent LLM-based validation cycle in mitigating logical hallucinations and ensuring the functional validity of the generated TCs within such a framework?

2 Related Work

The application of Natural Language Processing (NLP) techniques to TC generation has been extensively studied over the last decade [1, 14], and with the emergence of LLMs, test artifacts can be produced directly from textual requirements with reduced human effort [2]. However, these works largely assume access to well-structured requirements documents, leaving open the more challenging problem of extracting testable behaviors from *unstructured* mobile user manuals combining prose, lists, and embedded UI screenshots.

The limitations of single-model pipelines have motivated multi-agent designs in which specialized agents collaborate through structured roles [8]. In the software testing domain, agentic LLM-based frameworks have been applied to Behavior-Driven Development [13] and model-based test maintenance [10]. The closest related work introduces a multi-agent pipeline for rewriting and standardizing *existing* TC compo[3]. While that pipeline focuses on refining artifacts that already exist, our framework addresses the upstream problem of *generating* structured TCs from raw, visually rich PDF manuals, a setting that has not been explored by previous multi-agent designs.

Logical hallucinations remain a central failure mode of LLM-generated software artifacts [7], and recent large-scale studies on prompt engineering for unit test generation report that these issues persist even under carefully designed prompts [12]. Our work addresses this vulnerability from an architectural perspective rather than solely through prompt tuning, delegating verification to an independent and more capable judge agent.

3 Multi-Agent Pipeline

Given a PDF user manual M describing an interactive system such as a mobile device, the objective is to automatically produce a set of standardized TCs $\mathcal{T} = \{t_1, \ldots, t_n\}$ such that each t_i conforms to the five-component schema introduced in Section 1, is traceable to a UC explicitly documented in M, and is independently executable by a QA tester without further consultation of M. Three properties of real-world manuals render the task non-trivial: *multimodality*, since procedures combine narrative text with UI screenshots and iconography that are essential to disambiguate references such as *"Tap the share icon"*; *length*, with typical device manuals spanning hundreds of pages and exceeding the context window or fidelity envelope of single-agent LLM pipelines [11]; and *volatility*, as manuals are frequently revised for new product versions, leaving previously hand-authored TC suites outdated.

AUTOMMAS addresses these requirements through a two-phase multi-agent pipeline, illustrated in Figure 1. In the first phase, devoted to *UC identification*, a multimodal extraction agent processes the manual page by page and distills each documented procedure into a structured UC. In the second phase, devoted to *TC generation*, an iterative loop between a TC Generation Agent and an independent TC Validation Agent refines the TC derived from each UC until it satisfies a defined set of structural and semantic criteria. The Validator operates as an LLM-as-a-judge [18] positioned as a stronger reviewer than the Generator. Figure 2 illustrates the end-to-end output produced by AUTOMMAS, showing a manual page alongside its extracted UC and the resulting TC.

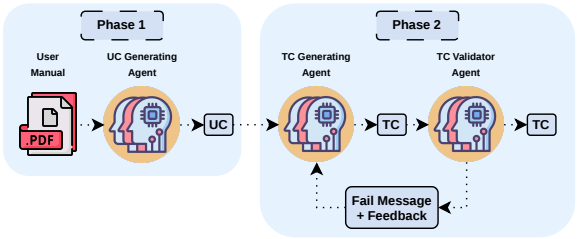


Figure 1: Architectural overview of AUTOMMAS.

3.1 System Architecture

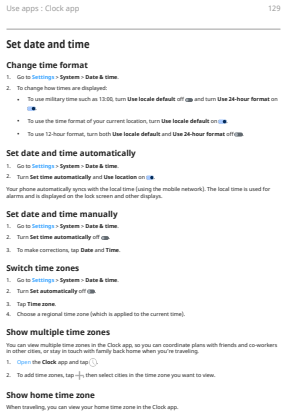
Each agent is assigned a model tailored to the cognitive demands of its role. The Extraction Agent requires multimodal perception to interpret screenshots, iconography, and callout arrows; therefore, it employs the `Qwen2.5-VL-7B vision`–language model. Its compact 7B scale keeps inference cost-effective for manuals spanning several hundred pages on a single GPU. The Generation and Validation agents operate exclusively on structured textual data and therefore use 70B text-only models under `Q4_K_M` quantization. The two roles differ in their underlying base models: the Validator runs the newer `Llama-3.3-70B`, a stronger instruction-following model than the Generator's `Llama-3.1-70B`, so each generated TC is reviewed by a more capable and independent model than the one that produced it. Table 1 summarizes the model assignments and the corresponding hyperparameters. Temperature is fixed at zero for all agents to enforce deterministic decoding, and token budgets correspond to the maximum artifact lengths observed in the development set.

Table 1: Models and generation hyperparameters per agent.

Agent	Model	Hyperparameters
Extraction	Qwen2.5-VL-7B	*temp* = 0; *max_tokens* = 1024
TC Generation	Llama-3.1-70B	*temp* = 0; *max_tokens* = 1024; *n_ctx* = 8192
TC Validation	Llama-3.3-70B	*temp* = 0; *max_tokens* = 512; *n_ctx* = 8192

3.2 Phase 1: Use Case Identification

The Extraction Agent processes the PDF page by page, sending each page to the `Qwen2.5-VL-7B-Instruct-Q4_K_M` MLLM introduced in Section 3.1. The model receives a system prompt that defines it as a QA automation specialist and a user prompt requesting a structured list of all complete procedures present on the page, each entry containing the *name*, *description*, and ordered *steps*. The page-level granularity constrains each inference call to a bounded context, mitigating the context degradation pathologies documented in LLM architectures applied to long documents [11]. If a page triggers an



(a) User Manual page

Field	Content
Name	Connect to a hidden Wi-Fi network
Description	Procedure to manually configure and connect the device to a hidden Wi-Fi network.
Steps	1. Open quick settings, then touch & hold the Wi-Fi icon. 2. Below the list of networks, tap Add network. 3. Enter the SSID, security type, and password. 4. Tap Advanced options and set Hidden network to Yes. 5. Tap Save. 6. Select the saved hidden network and tap Connect if required.

(b) Extracted Use Case

Component	Content
Title	TC - Connect to Hidden Wi-Fi Network
Summary	Verify that the user can connect the device to a hidden Wi-Fi network using the Settings menu.
Initial Setup	- Wi-Fi is enabled. - No hidden Wi-Fi network is connected.
Steps	1. Open quick settings, then touch & hold the Wi-Fi icon. 2. Below the list of networks, tap Add network. 3. Enter the SSID (network name), security type, check Show password and enter the password. 4. Tap Advanced options, and change Hidden network to Yes. 5. Tap Save. 6. If you were connected to Wi-Fi when you added the hidden network, in the list of available networks, tap the name of the saved network, then tap Connect.
Expected Results	- The device should connect to the hidden Wi-Fi network. - The hidden network should appear in the list of available networks.

(c) Generated Test Case

Figure 2: End-to-end example of AUTOMMAS. (a) A page from the User Manual; (b) the UC extracted; (c) the TC generated.

out-of-memory error on the GPU, the agent falls back to a text-only inference with a reduced token budget.

A post-processing stage then eliminates redundant extractions: UC names are normalized with respect to letter casing, an alias table curated from the development set and available in the artifact repository maps variants of the same procedure to a single canonical name, and, within each duplicate group, the entry with the largest number of extracted steps is retained, with ties broken by description length. The output is a canonical and non-redundant set of UCs.

3.3 Phase 2: Test Case Generation

Phase 2 receives the canonical set of UCs produced by Phase 1 and generates a complete and standardized TC for each entry through an iterative multi-agent loop comprising a TC Generation Agent and a TC Validation Agent.

3.3.1 TC Generation Agent. The TC Generation Agent, powered by the `Llama-3.1-70B-Instruct-Q4_K_M.gguf` model, receives a UC and produces a complete TC covering the five components introduced in Section 1 (Title, Summary, Initial Setup, Steps, Expected Results). Its prompt encodes (i) the syntactic and semantic rules for each component; (ii) the entity-modality-outcome formulation required for Expected Results [15]; and (iii) an atomicity constraint mandating that each TC exercise exactly one functional scenario. The generated TC is forwarded to the TC Validation Agent.

3.3.2 TC Validation Agent. The TC Validation Agent, powered by the `Llama-3.3-70B-Instruct-Q4_K_L.gguf` model, receives the originating UC and the candidate TC, and emits a structured PASS/FAIL verdict for each of the four criteria, in addition to an aggregated *Overall* verdict and a short actionable feedback paragraph. Specifically, the agent verifies, for *Completeness*, that all five components are present and non-empty; for *Atomicity*, that the TC exercises exactly one behavior and each Step describes a single tester action; for *Clarity*, that each Step and Expected Result is unambiguous, independently executable, and follows the entity-modality-outcome pattern [15]; and for *Traceability*, that the TC references the same functional objective as the originating UC without introducing artificial UI elements or outcomes. The Overall verdict is the logical conjunction of the four criteria. The complete validation prompt is available in the artifact repository.

If the Overall verdict is PASS, the TC is accepted as final. Otherwise, a feedback paragraph specifying the violated criteria and the required corrections is appended to the generation context, and the TC Generation Agent produces a revised version in the next iteration. The cycle repeats for a maximum of 20 iterations per TC, after which the latest generated version is retained and flagged for manual review; this cap bounds the worst-case cost per TC and, as Section 5 shows, is not restrictive in practice, since all TCs converge below it. This separation between generation and validation responsibilities follows the reflective validation strategy proposed by [3].

4 Experimental Setup

We evaluate the proposed pipeline using the official user manual of the Motorola Edge 50 Neo (Global, EN-US edition)[1], a publicly available PDF document of 379 pages covering the complete functional workflows of the device. The manual is partitioned at the page level through a stratified 70/30 split into a development set, used during

[1] https://en-us.support.motorola.com/app/answers/detail/a_id/179728/

pipeline construction and prompt refinement, and a held-out evaluation set, used exclusively for the experiments reported in Section 5. A lightweight filter removes non-instructional pages, namely the cover, table of contents, indexes, and any page with fewer than 150 characters of extractable text, since these contain no actionable procedure. After filtering, the development set comprises 260 pages and the evaluation set 108 pages. This partitioning and page filtering belong to the experimental design of this study and are not required steps of the pipeline, which processes arbitrary manuals end to end. To keep the blinded four-expert protocol feasible, since each expert independently judges all 210 resulting artifacts, a random sample of 30 pages is drawn from the evaluation set; these 30 pages are processed by every Phase 1 system to produce the UCs evaluated by QA experts, and the UCs extracted by AUTOMMAS on these same pages are subsequently passed to every Phase 2 system to produce the TCs evaluated in the blinded pool.

4.1 Evaluation Metrics

Following the absence of a verifiable ground truth for UCs in unstructured user manuals, the evaluation protocol combines expert human judgment with automated quantitative metrics.

Phase 1. each extracted UC u receives a binary label (correct/incorrect) from each of the four QA evaluators. Let $c(u)$ be the number of evaluators who labeled u as correct. A UC is accepted under majority voting when $c(u) \geq 3$. The *Extraction Validity Rate* (EVR) is defined as $\text{EVR} = \frac{1}{|U|} \sum_{u \in U} \mathbb{1}[c(u) \geq 3]$, where U denotes the set of UCs extracted by the system under evaluation, i.e., the fraction of extracted UCs accepted by a majority of evaluators, reported with 95% Wilson confidence intervals [16].

Phase 2. TCs are assessed at two complementary levels. At the *pipeline level*, three automated metrics characterize the Generator–Validator loop: the *Validator Acceptance Rate*, the proportion of TCs whose final version receives a PASS verdict from the LLM-as-a-judge Validator; the *Convergence Rate*, the proportion of TCs reaching acceptance within the 20-iteration cap and therefore not flagged for manual review; and the *Average Iterations*, the mean number of refinement cycles required until acceptance. The Validator Acceptance Rate reflects the automatic judgment and is distinct from the human-assigned TCA defined below.

At the *human level*, QA experts evaluate each TC under a binary (*I agree*/*I don't agree*) scale, both globally and per criterion. The global judgment yields the *TC Acceptance* (TCA) rate, the proportion of TCs accepted as a whole. The per-criterion evaluation covers four dimensions: *Completeness*, indicating whether all five TC components are present and well-formed; *Atomicity*, indicating whether the TC evaluates a single functional behavior; *Clarity*, assessing whether the Steps and Expected Results are unambiguous and independently executable; and *Traceability*, verifying whether the TC maintains an explicit link to its originating UC and corresponding section of the manual. All human metrics are aggregated by majority voting, and agreement among evaluators is reported as the percentage of unanimous judgments.

4.2 Baselines

Phase 1 Baselines. To isolate the contribution of the multimodal extraction strategy, AUTOMMAS is compared with three Phase 1 baselines. The first is a *text-only* pipeline composed of EasyOCR for optical character recognition coupled with `Llama-3.1-70B`, operating on the extracted OCR text under the same system and user prompts employed by the multimodal extraction agent. Using the same backbone as the TC Generation Agent ensures that any observed difference is attributable to the extraction modality rather than to the language model. The remaining two baselines are commercial frontier systems Gemini 3.5 Flash and GPT-5.5 Instant which receive the manual pages directly and are prompted to return the same structured UC schema.

Phase 2 Baselines. For TC generation, AUTOMMAS is compared with two frontier LLMs operating under a zero-shot configuration: Gemini 3.5 Flash and GPT-5.5 Instant. Each baseline receives the same UC input and prompt template used by the TC Generation Agent, but generates a single TC in a one-shot manner, without iterative validation or corrective feedback loops. Comparing with frontier zero-shot models isolates the contribution of the iterative multi-agent architecture from the intrinsic generative capability of the underlying language models.

4.3 Manual Evaluation Protocol

The blinded evaluation pool comprises two independent pools, one per pipeline phase. For Phase 1, the 30 sampled pages produce a total of $30 \times 4 = 120$ UC candidates across AUTOMMAS, the OCR+LLM baseline, Gemini 3.5 Flash and GPT-5.5 Instant. For Phase 2, the 30 UC candidates extracted by AUTOMMAS are passed to three TC generators: AUTOMMAS itself, Gemini 3.5 Flash, and GPT-5.5 Instant, yielding a total of $30 \times 3 = 90$ TCs. All artifacts are presented to four QA experts within randomized and blinded pools, omitting any information regarding the originating system. Each expert independently evaluates every artifact according to the qualitative metrics defined in Section 4.1. The final label assigned to each artifact is determined through majority voting.

5 Results

We organize and present the results around the two pipeline phases: Use Case Extraction, which addresses **RQ1**, and Test Case Generation, which together with a Validator–Human gap analysis addresses **RQ2**.

5.1 Use Case Extraction

AUTOMMAS achieves an Extraction Validity Rate (EVR) of 87%, and three observations frame this result. First, multimodal extraction decisively outperforms the text-only approach: AUTOMMAS surpasses the OCR+LLM baseline by 11 pp while using a 7B model, isolating the gain to modality rather than language model scale. Second, despite relying on a substantially smaller locally deployed 7B multimodal model, AUTOMMAS slightly outperforms the commercial frontier system Gemini 3.5 Flash, which achieved an EVR of 86%.

Third, only GPT-5.5 Instant attains a higher EVR of 93%; however, it depends on a commercial API with higher per-token cost,

whereas AUTOMMAS runs on a single GPU without usage limits or data privacy exposure, a relevant advantage when manuals contain confidential product information. The confidence intervals of AUTOMMAS, Gemini 3.5 Flash, and GPT-5.5 Instant substantially overlap, so the EVR ranking among these three systems should be interpreted as indicative rather than definitive given the current sample size, as shown in Table 2.

Regarding inter-rater consistency, AUTOMMAS achieved a 64% unanimous agreement rate, indicating that its output does not introduce greater evaluator disagreement when compared to commercial systems such as Gemini 3.5 Flash and GPT-5.5 Instant. The OCR+LLM baseline, with a unanimity rate of 33%, systematically produces ambiguous or borderline UCs, indicating that the inferiority of the unimodal approach extends beyond what EVR alone suggests.

Table 2: Phase 1: Use Case extraction accuracy per system. EVR is the proportion of extracted use cases judged correct by majority vote of four QA evaluators.

System	EVR	95% CI	% Unanimous
Gemini 3.5 Flash	86%	[70, 94]	63%
GPT-5.5 Instant	93%	[79, 98]	66%
AUTOMMAS	87%	[70, 95]	64%
OCR+LLM	76%	[59, 88]	33%

5.2 Test Case Generation

As can be seen in Figure 3a, Phase 2 provides the most conclusive evidence supporting the proposed architecture. AUTOMMAS achieves a TCA of 97%, while Gemini 3.5 Flash and GPT-5.5 Instant achieve 93% and 73%, respectively. This difference is notable given that these are widely deployed commercial models, optimized for latency and cost, operating under the same prompt template as the AUTOMMAS generation agent and differing only in the absence of an iterative correction loop. This result constitutes direct evidence that the multi-agent validation architecture provides a quality advantage that the intrinsic generative capability of a single-step model cannot replicate.

Inter-rater consistency reinforces this difference: AUTOMMAS reaches 63% unanimous agreement among the four evaluators, compared with 42% for Gemini 3.5 Flash and 40% for GPT-5.5 Instant, as reported in Table 3. The higher unanimity indicates that the TCs accepted from AUTOMMAS are not only approved more frequently, but are also less ambiguous to evaluators.

The internal dynamics of the Generator–Validator loop are presented in Figure 3b. Approximately 50% of the TCs are accepted in Iteration 1 without refinement, while the remaining cases enter the feedback loop and are rapidly accepted throughout Iterations 2–10 as the Generator incorporates the Validator's feedback. All TCs converge before Iteration 18, well below the 20-iteration cap, so most of the quality improvement concentrates in the early iterations while the loop cost remains bounded.

Table 3: Phase 2: TC acceptance per system. TCA is the proportion of generated TCs accepted under a global judgment by majority vote of QA evaluators.

System	TCA	95% CI	% Unanimous
AUTOMMAS	**97%**	**[83, 99]**	**63%**
Gemini 3.5 Flash	93%	[79, 98]	42%
GPT-5.5 Instant	73%	[56, 86]	40%

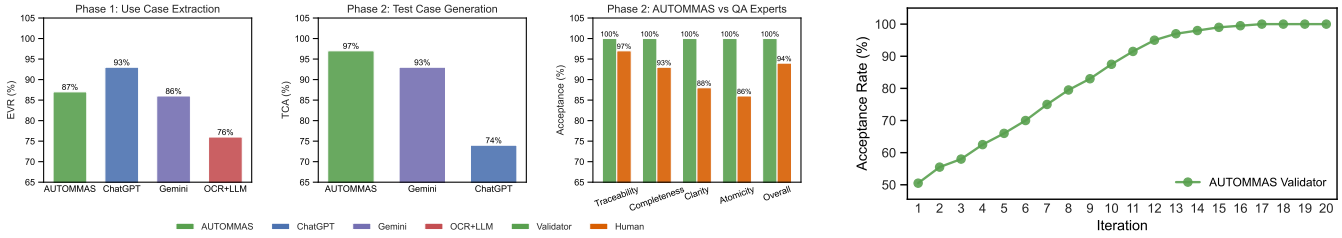
5.3 Validator-Human Gap Analysis

The criterion-level comparison between the LLM-based Validator and the human evaluators reported in Table 4 reveals a systematic increase in disagreement as the subjective complexity of the criterion grows. Traceability presents the smallest gap of 3.3 pp, indicating that the semantic correspondence between the generated TC and the source UC can be automatically validated with high reliability. In contrast, the gaps increase to 10.0 pp for Completeness, 13.3 pp for Clarity, and 16.7 pp for Atomicity, showing that the Validator tends to accept structurally correct or linguistically fluent TCs even when human experts consider them insufficiently precise or functionally over-composed. The largest discrepancy in Atomicity demonstrates that determining whether a TC exercises exactly one functional behavior remains a challenge for LLM-based validation without stronger negative examples or expert-guided constraints. Overall, the residual global gap of 6.7 pp between automatic and human validation suggests that fully unsupervised deployment may still produce false positives in complex scenarios.

Regarding evaluation time, Table 5 reveals that human assessment of the Phase 2 outputs remained relatively similar across systems, with Gemini 3.5 Flash requiring 25.9 s, AUTOMMAS 27.4 s, and GPT-5.5 Instant 31.9 s per TC. The longer evaluation time for GPT-5.5 Instant correlates with its lower TCA of 73%, since identifying and justifying defects requires greater cognitive effort than validating correct outputs. In the gap analysis, AUTOMMAS automatic validation achieved a median time of 8.7 s compared to 31.7 s for human evaluation, representing a 3.6× acceleration. While human evaluation time increases with annotation complexity and procedural difficulty, particularly for advanced workflows such as biometric enrollment or security settings, the automatic Validator maintains constant execution time regardless of TC complexity, eliminating fatigue and inter-evaluator variability.

Table 4: Gap analysis: per-criterion acceptance of AUTOMMAS TCs by human evaluators, compared with the LLM Validator. The gap is expressed in percentage points (pp).

Criterion	AUTOMMAS	Manual	95% CI	Gap (pp)
Traceability	100%	97%	[83, 99]	3.3
Completeness	100%	90%	[74, 97]	10.0
Clarity	100%	87%	[70, 95]	13.3
Atomicity	100%	83%	[66, 93]	16.7
Overall	100%	93%	[79, 98]	6.7



(a) Human evaluation of the two pipeline phases.

(b) Validator acceptance per iteration.

Figure 3: Evaluation results of AUTOMMAS.

Table 5: Median time per TC (seconds). TC Evaluation reports the human evaluation time for each system. The gap analysis compares, for AUTOMMAS only, human evaluation time versus automatic validation time.

	TC Evaluation	Gap Analysis (AUTOMMAS)	
System	**Manual (s)**	**Manual (s)**	**Automatic (s)**
Gemini 3.5 Flash	25.9s	—	—
AUTOMMAS	27.4s	31.7s	**8.7s**
GPT-5.5 Instant	31.9s	—	—

6 Limitations and Threats to Validity

Threats to validity were structured following the classification proposed by Wohlin et al. [17]. In terms of internal validity, the primary threat concerns the consistency of human evaluation, since, although the criteria were formally defined, their application still requires qualitative judgment, particularly for Atomicity and Clarity. Unanimity rates between 40% and 63% reveal disagreements among evaluators in a substantial fraction of the analyzed TCs. Additionally, evaluator fatigue may have affected attention during later stages of the process.

Regarding external validity, the evaluation was limited to a single user manual belonging to an Android device and written in English, which restricts the generalization of the results to other contexts, domains, or languages. Two architectural limitations also constrain the current results. The pipeline enforces a one-to-one mapping between UCs and TCs, whereas standard test design derives multiple TCs from a single UC to cover alternative and exception flows; the generated suites may therefore omit such scenarios, and the reported metrics do not measure functional coverage over the manual as a whole. In addition, UCs hallucinated during Phase 1 propagate to Phase 2, where the Validator, which does not access the source manual, cannot detect them. Concerning construct validity, the EVR and TCA metrics rely on expert judgment due to the absence of a formal ground truth. Finally, conclusion validity is limited by the sample size used in the evaluation.

7 Conclusions

This paper presented AUTOMMAS, a multi-agent pipeline for the automatic generation of TCs from unstructured mobile device PDF user manuals. Regarding **RQ1**, multimodal understanding of UI screenshots and graphical annotations proved essential for accurate extraction, as demonstrated by the 11 pp EVR advantage over the OCR+LLM baseline obtained with a substantially smaller 7B model. Regarding **RQ2**, iterative corrective validation constitutes the main architectural factor driving quality: AUTOMMAS achieved a 23 pp TCA advantage over `GPT-5.5 Instant` under the same prompt template, with approximately 50% of the TCs accepted without refinement and the remaining cases converging in early iterations, while automatic validation ran 3.6× faster than human evaluation with constant execution times regardless of TC complexity. These findings are scoped to the evaluated context of a single English-language Android device manual.

In practical terms, the generated TCs are documentation-level artifacts intended for direct manual execution by QA testers and as a structured intermediate representation for subsequent test script automation. Automatic validation still diverges from expert judgment by 6.7 pp overall, with the largest discrepancy in Atomicity: many manual pages describe procedures without delimiting where one functional behavior ends and the next begins, leading the extraction agent to merge adjacent actions into UCs that propagate into non-atomic TCs, which the Validator, lacking an external reference for behavioral granularity, accepts as structurally fluent. As future work, we plan to evaluate generalization across manuals from other manufacturers, platforms, and languages; to assess additional model configurations for each agent role, both as replacements and as complementary alternatives; to relax the one-to-one UC–TC mapping toward covering alternative and exception flows; and to execute the generated TCs in device automation environments, measuring how structural quality relates to runtime execution behavior.

Artifact Availability

The authors declare that the research artifacts supporting the findings of this study, including the experimental data, the prompts of all agents, the source code, and the results, are available at: https://github.com/AndresDavidPeralta/AUTOMMAS.

Acknowledgements

This work was supported by the following Brazilian agencies: the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES-PROEX)- Finance Code 001, the National Council for Scientific and Technological Development (CNPq) through projects CNPq - 313137/2025-0; and CNPq 406506/2025-6, and Amazonas State Research Support Foundation- FAPEAM- through the POS-GRAD project 2024/2025, and by Motorola Mobility Comércio de Produtos Eletrônicos Ltda, under the terms of Federal Law No. 8.387/1991, through agreement No. 02/2021 with ICOMP/UFAM.

References

[1] M. R. Arya Devi and P. Abdul Jabbar. 2026. Automated Test Case Generation Using Natural Language Processing. In *ICT for Intelligent Systems: Proceedings of the International Conference on ICT for Intelligent Systems (ICTIS 2025)*, Jyoti Chaudri, Parikshit N. Mahalle, Thinagaran Perumal, and Amit Joshi (Eds.). Springer Nature Singapore, Singapore, 127–138. doi:10.1007/978-981-96-8901-9_12

[2] Mohammad Baqar and Rajat Khanda. 2025. The Future of Software Testing: AI-Powered Test Case Generation and Validation. In *Intelligent Computing: Proceedings of the Computing Conference (CompCom 2025) (Lecture Notes in Networks and Systems, Vol. 1424)*, Kohei Arai (Ed.). Springer, Cham, Switzerland, 276–300. doi:10.1007/978-3-031-92605-1_18

[3] Gabriel Carvalho, Andrés Peralta, André Carvalho, Yan Soares, Igor Lima, Nikson Ferreira, and Hallyson Melo. 2025. ARTeMIS: Agent-based Rewriting and Test Case Management with Intelligent Supervision. In *Anais do XXIV Simpósio Brasileiro de Qualidade de Software*. SBC, Porto Alegre, RS, Brasil, 193–203. doi:10.5753/sbqs.2025.15058

[4] Mouna Hammoudi, Gregg Rothermel, and Andrea Stocco. 2016. WATERFALL: An Incremental Approach for Repairing Record-Replay Tests of Web Applications. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. Association for Computing Machinery, Seattle, WA, USA, 751–762. doi:10.1145/2950290.2950294

[5] IEEE. 2008. IEEE Standard for Software and System Test Documentation. *IEEE Std 829-2008 (Revision of IEEE Std 829-1998)* 829-2008 (2008), 1–150. doi:10.1109/IEEESTD.2008.4578383

[6] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard. 1992. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, Reading, MA.

[7] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* 55, 12, Article 248 (2023), 38 pages. doi:10.1145/3571730

[8] Cristian Jimenez-Romero, Alper Yegenoglu, and Christian Blum. 2025. Multi-agent systems powered by large language models: applications in swarm intelligence. *Frontiers in Artificial Intelligence* 8 (2025), 1–23. doi:10.3389/frai.2025.1593017

[9] Maurizio Leotta, Andrea Stocco, Filippo Ricca, and Paolo Tonella. 2016. ROBULA+: An Algorithm for Generating Robust XPath Locators for Web Testing. *Journal of Software: Evolution and Process* 28, 3 (2016), 177–204. doi:10.1002/smr.1771

[10] Carlos D. Q. Lima, Everton L. G. Alves, and Wilkerson L. Andrade. 2024. A Systematic Literature Review on MBT Test Cases Maintenance. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, Osaka, Japan, 1356–1365. doi:10.1109/COMPSAC61105.2024.00179

[11] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173. doi:10.1162/tacl_a_00638

[12] Wendkûuni C. Ouédraogo, Abdoul Kader Kaboré, Yinghua Li, Haoye Tian, Anil Koyuncu, Jacques Klein, David Lo, and Tegawendé F. Bissyandé. 2026. Prompt Engineering in LLMs for Automated Unit Test Generation: A Large-Scale Study. *Empirical Software Engineering* 31, 4, Article 103 (2026). doi:10.1007/s10664-026-10840-4

[13] Ciprian Paduraru, Miruna Zavelca, and Alin Stefanescu. 2025. Agentic AI for Behavior-Driven Development Testing Using Large Language Models. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*. INSTICC, SciTePress, Setúbal, Portugal, 805–815. doi:10.5220/0013374400003890

[14] Srinivas Perala and Ajay Roy. 2021. A Review on Test Automation for Test Cases Generation using NLP Techniques. *Turkish Journal of Computer and Mathematics Education* 12, 6 (2021), 1488–1491.

[15] Roelf J. Wieringa and Anne Persson. 2010. *Requirements Engineering: Foundation for Software Quality*. Vol. 6182. Springer, Berlin, Germany. 12 pages.

[16] Edwin B. Wilson. 1927. Probable Inference, the Law of Succession, and Statistical Inference. *J. Amer. Statist. Assoc.* 22, 158 (1927), 209–212.

[17] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2000. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Boston, MA, USA.

[18] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 36. 46595–46623.