How Software Testers Perceive Large Language Models: A Focus Group Study with Lexicometric Analysis

Izabella Silva
Federal University of Campina Grande (UFCG)
Campina Grande, Brazil
izabella.silva@virtus.ufcg.edu.br

Mirko Perkusich
Federal University of Campina Grande (UFCG)
Campina Grande, Brazil
mirko@virtus.ufcg.edu.br

Danyllo Albuquerque
Federal University of Campina Grande (UFCG)
Campina Grande, Brazil
danyllo.albuquerque@virtus.ufcg.edu.br

Kyller Gorgônio
Federal University of Campina Grande (UFCG)
Campina Grande, Brazil
kyller@virtus.ufcg.edu.br

Angelo Perkusich
Federal University of Campina Grande (UFCG)
Campina Grande, Brazil
perkusich@virtus.ufcg.edu.br

ABSTRACT

Context. Large Language Models (LLMs) are increasingly embedded in software-engineering toolchains, with potential to enhance quality-assurance tasks such as test-case generation, bug analysis, and traceability. Yet their *industrial* value remains unclear, partly because practicing testers' perspectives are underexplored. **Objective**. We investigate how industry testers perceive the usefulness and limitations of LLMs across the Software Test Life Cycle (STLC), identifying (i) the most critical stages, (ii) operational bottlenecks, and (iii) perceived benefits, risks, and adoption conditions. **Method**. A focus group with five professional quality analysts (2–10 years' experience) included hands-on interaction with an LLM-powered BDD notebook, followed by guided discussion and affinity-diagram voting. Forty-two statements were coded and prioritized. **Results**. *Requirements Analysis* and *Test Planning* were deemed most critical; 71% of votes linked rework to unclear or incomplete requirements. Benefits cited were automation of repetitive tasks, broader coverage, and faster learning; main concerns were prompt sensitivity, limited domain generalization, and data-privacy risks, with emphasis on human oversight and domain-adapted prompt libraries. **Conclusion**. While LLMs can improve efficiency and coverage, adoption depends on high-quality inputs, secure deployment, and early-phase integration. The findings offer empirically grounded guidance for aligning LLM solutions with the socio-technical realities of industrial testing teams.

KEYWORDS

Large Language Model; Software Testing; Software Test Life Cycle; Prompt Engineering; Requirements Analysis.

1 Introduction

Software testing is a cornerstone of software quality assurance [13], but also one of its most resource-intensive activities [2, 5]. Its execution is structured by the *Software Test Life Cycle* (STLC)[4], which organizes key phases such as requirements analysis, planning, design, execution, and closure. However, practitioners routinely face persistent obstacles—ambiguous requirements, fragile automation frameworks, and fragmented tooling—that undermine effectiveness and cause early defects to cascade downstream[2, 3, 9]. These limitations highlight the need for techniques that reduce manual effort while increasing fault detection and test coverage.

In recent years, Large Language Models (LLMs), such as GPT-4 and PaLM 2, have emerged as promising tools to support software engineering tasks through natural-language reasoning and code synthesis [13]. In the context of testing, LLMs have demonstrated potential in automating tasks such as test-case generation, test repair, and bug reproduction [1, 12, 14]. Preliminary empirical studies confirm their technical promise in both academic and industrial settings [7]. However, most evaluations remain laboratory-based and code-centric, leaving a critical gap: how do practicing software testers perceive the usefulness, risks, and adoption conditions of these tools within real-world testing workflows?

This study addresses that gap by shifting the focus from model capabilities to practitioner experience. Specifically, we investigate how industry testers view LLM-based tools across the STLC, including which phases they consider most critical, what obstacles they encounter, and how they evaluate the trade-offs of LLM adoption.

To this end, we conducted an exploratory focus group with five professional quality analysts. Participants first interacted with an LLM-powered BDD notebook and subsequently engaged in a structured discussion and affinity-voting exercise to prioritize their observations. To enhance analytical depth and reliability, we employed a mixed-methods approach combining qualitative coding, statistical aggregation of votes and polls, and lexicometric analysis of the transcribed session using the IRaMuTeQ software.

Our study yields four key contributions:(i) A qualitative account of testers' perceptions of LLM applicability and trustworthiness across STLC phases; (ii) An empirically grounded ranking—supported by voting data and lexical clustering—that identifies *Requirements Analysis* and *Test Planning* as the most critical stages; (iii) A mapped set of perceived benefits (e.g., efficiency, coverage, upskilling) versus limitations (e.g., prompt sensitivity, domain gaps, data privacy), with thematic nuances extracted from the corpus; and (iv) Practical guidance for researchers and vendors on aligning LLM-based solutions with the socio-technical realities of QA teams. The remainder of this paper is organized as follows. Section 2 details the methodological procedures. Section 3 presents the empirical findings and discusses them in light of the research questions. Section 4 concludes with implications, limitations, and directions for future work.

2 Study Settings

This section outlines the methodological framework used to investigate practitioners' perceptions of LLMs across the STLC. Our approach comprised four stages described in the following.

Stage 1 - Protocol Definition. Following established practices in qualitative software engineering research [8, 10], we designed a study to explore how professional testers perceive the applicability of LLMs throughout the STLC. The study addressed three research questions:

- **RQ1**: Which STLC phases do testers consider most critical, and what obstacles make those phases particularly challenging?
- **RQ2**: What benefits do testers expect from integrating LLMs into their workflows?
- **RQ3**: What trade-offs do testers perceive when adopting LLM-based assistance in testing workflows?

RQ1 probes whether testers continue to prioritize early STLC phases, echoing the traditional view that early-stage defects drive downstream costs, especially in an era of LLM-augmented tools. RQ2 captures perceived benefits such as efficiency, test coverage, and learning support. RQ3 elicits concerns including prompt sensitivity, domain adaptation, and data privacy. These three questions shape the reporting of our findings.

To capture socially grounded experiences, we employed a focus group design—a well-established method in software engineering research for eliciting practitioner insights [6, 11]. The remote, semi-structured focus group was conducted via Microsoft Teams. Participants completed a demographic questionnaire beforehand. The session followed a semi-structured script that included ten STLC-aligned discussion prompts, Likert-style polls to collect quantitative data, and a final voting session using affinity diagramming. To ensure procedural rigor, three researchers piloted all materials internally before deployment. The protocol included a 12-item demographic survey distributed one week in advance, verbatim audio-video recording of the session, and immediate transcription for analysis. All volunteers signed informed-consent forms that covered recording, anonymization, and eventual public release of the anonymized dataset. To strengthen reliability, two researchers independently coded the transcript, maintained an audit trail of coding decisions, and conducted member checking by sending participants a one-page summary of preliminary themes for confirmation.

Stage 2 -Participants and Sampling. We used purposive sampling to recruit five senior quality analysts (2-10 years of experience) from a single industrial partner: a university-affiliated R&D center in Brazil. The organization has high process maturity, delivering a wide range of technology projects (e.g., AI, web, embedded systems), and widely employs Agile practices. The selected participants reported diverse experiences with testing tools and prior exposure to LLM-based chatbots. All participants provided written informed consent under institutional ethics approval.

Stage 3 - Focus Group Execution. We conducted a 90-minute remote session on Microsoft Teams. Participants first discussed challenges across the STLC, then shared specific benefits and concerns regarding LLM use. These discussions were supplemented by anonymous Likert-style polls to quantify perceptions. The session concluded with an affinity voting exercise on a Miro board, where 42 collected statements were clustered and prioritized by the group to produce a visual map of collective priorities and concerns. All materials were archived for analysis.

Stage 4 - Data Collection and Analysis. With participants' informed consent, we recorded audio, video, and shared-screen content during the focus group session. The session was transcribed and manually refined to ensure accuracy. All identifying information was removed to protect participant privacy. The resulting corpus, composed of 107 speech turns marked by "****", was translated into English (via Google Translate), encoded in UTF-8, and automatically segmented into 40-word windows.

We adopted a mixed-methods approach. Quantitative data—such as poll results and affinity-voting counts—were tabulated to support the prioritization of themes. For qualitative insights, we applied both thematic coding and a lexicometric analysis using IRaMuTeQ, which enabled the discovery of thematic classes based on statistical patterns in the vocabulary.

The IRaMuTeQ analysis was configured as follows (see Table 1): segments of 40 words; a minimum word frequency of 3; standard English lemmatization and stop-lists; and Chi-squared tests (χ^2, $p < 0.05$) to determine word-class associations. The core clustering technique was the Descending Hierarchical Classification (DHC) using Reinert's algorithm, with a maximum of 10 initial classes.

Table 1: IRaMuTeQ Analysis Processing Settings

Parameter	**Value**
Segment size	40 words
Minimum frequency (active forms)	3
Lemmatization	English (standard stemmer)
Stop-lists	Standard EN (functional words)
Association test	χ^2, $p < 0.05$
DHC Algorithm	Reinert (phase 1 max. = 10 classes)

This process yielded three main analytical artifacts:

- **Global Word Cloud:** A visualization of the 100 most frequent terms (min. freq. = 3), providing an initial overview of core vocabulary.
- **DHC Dendrogram:** A tree diagram representing the hierarchical structure of lexical classes extracted from the corpus.
- χ^2 **Tables (`.csv`):** For each class, a list of over-represented words with high statistical association, serving as input for semantic labeling and thematic interpretation.

To strengthen credibility, we triangulated the findings across three sources: (i) the quantitative polling and voting data, (ii) the lexicometric analysis outputs, and (iii) researchers' manual annotations. We also conducted *member checking* by sending participants a summary of preliminary themes for validation and alignment.

3 Results and Discussion

This section presents the main findings from the focus group study, based on a mixed-methods analysis that triangulated affinity voting, Likert-scale polls, and a lexicometric analysis of the transcribed discussion. We begin by outlining the participants' profile, which contextualizes their perspectives (section 4.1). Next, we report corpus-level statistics and the thematic classes identified through DHC (Section 4.2). These results provide the foundation for addressing the research questions (Section 4.3, 4.4, and 4.5), each discussed in light of both qualitative insights and supporting quantitative evidence. The section closes with an integrated synthesis and implications (Section 4.6). To facilitate replication and secondary analysis, all supplementary materials—including survey spreadsheets, the anonymized transcript, Miro board exports, and analysis scripts—are available online at [1].

3.1 Participant Profile and Contextual Framing

The focus group session took place in May 2025 and lasted 90 minutes. It comprised five experienced quality assurance professionals drawn from a Brazilian R&D center with high process maturity and diversified project portfolios. All participants held mid- to senior-level positions and had practical exposure to Agile-based development and testing workflows. Their combined expertise provided a solid basis for critically evaluating the integration of LLMs into the Software Test Life Cycle (STLC).

Table 2 summarizes the key demographic and technical attributes that contextualize their viewpoints. While all participants reported extensive QA experience (median = 7 years), their exposure to LLMs in testing tasks varied significantly, ranging from exploratory to

[1]Supplementary Material

moderately hands-on. This combination of deep domain knowledge and emerging familiarity with LLM-based tooling positioned the group to articulate both the potential and the operational constraints of such technologies in practice.

Table 2: Profile of focus group participants ($N = 5$)

Attribute	Distribution
Age band	1 @ 24–29; 3 @ 30–35; 1 @ 36–41
Seniority	2 Specialist/Lead; 1 Master; 1 Senior; 1 Mid-level
Years in QA	All >5 years (median = 7)
Familiarity with BDD	2 basic; 3 advanced (frequent users)
BDD tooling	Cucumber (3), Behave (1), none (1); one noted past use of RobotFramework
Prior LLM use in testing	Test-case writing (2), automation (2), code smell detection (1), test-smell detection (1), and data generation (1); two had no test-related LLM use
Overall LLM experience	1 very positive, 1 positive, 3 never used in testing context
LLM tools tried	ChatGPT (4), Gemini (1), DeepSeek (1), Copilot (1)

This configuration ensured a heterogeneous yet qualified panel, capable of offering grounded assessments of LLM utility, usability, and limitations within typical software testing workflows.

3.2 Descriptive Results and Thematic Classes

This subsection details the results of our lexicometric analysis, outlining the corpus statistics, the global word cloud, and the profiles of three thematic classes derived from the DHC algorithm. This analysis provides the quantitative and thematic foundation for discussing our research questions.

General Corpus Statistics. The general statistics of the analyzed corpus, detailing its size and composition, are summarized in Table 3.

Table 3: General Statistics of the Analyzed Corpus

Metric	Value
Texts (blocks)	107
Segments (40 tokens)	143
Occurrences (tokens)	3,041
Forms (types)	843
Lemmas	753
Active forms	490
Lexical classes	3
Classified segments	52 / 143 (36.4%)

As the table indicates, the corpus is composed of 107 participant statements, providing a solid foundation for the qualitative analysis. The key finding from this initial statistical overview is the identification of **3 distinct lexical classes**, which form the basis for the thematic analysis presented in the following sections. It is also noteworthy that the classification algorithm was able to assign a class to **36.4%** of the text segments. While this percentage confirms the stability and distinctiveness of the identified themes, it also suggests considerable lexical heterogeneity in the remainder of the corpus, a point that will be revisited in the discussion of the study's limitations.

Word-Cloud global. To provide an initial high-level overview of the participants' discourse, a word cloud was generated from the corpus, as shown in Figure 1. In this visualization, the size of each word is proportional to its frequency in the text, highlighting the most central concepts discussed during the focus group.

A qualitative analysis of the word-cloud reveals that the terms *test* and *requirement* are the most prominent, confirming their

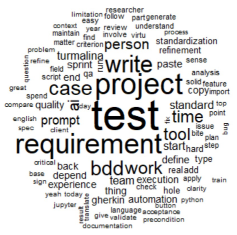

Figure 1: Frequent terms in the corpus.

central role in the discussion. Their dominance suggests they are transversal concepts that underpin nearly all topics addressed by the participants, from planning to execution.

Thematic Classes from DHC. The primary method for thematic analysis was the DHC, which partitioned the classified text segments into three stable classes. The hierarchical relationship between these classes is visualized in the dendrogram presented above (Figure 2.

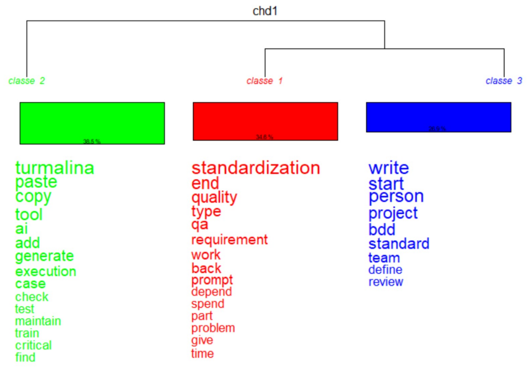

Figure 2: Dendrogram of the DHC analysis.

The structure of the dendrograms reveals the architecture of the participants' discourse. The first major split separates Class 2 (Testing & Automation) from the other two classes. This indicates that the discussion around automation and tools constitutes the most distinct and cohesive lexical universe for the group. The other main branch of the dendrogram groups Class 1 (Quality & Standardisation) and Class 3 (Specification & BDD) together, showing that conversations about quality, standards, and specifications are closely interconnected. Together, they form a broader super-theme focused on the "foundations of the testing process". A detailed profile of each of these three classes is presented next.

Lexical Profile of the Thematic Classes. The semantic profile of each of the three classes is detailed in Table 4. This table presents the top five keywords for each class, ranked by their Chi-squared (χ^2) test value. The χ^2 value measures the strength of association between a word and a specific class; a higher value indicates a more characteristic relationship, signifying that the word is over-represented in that class compared to the rest of the corpus.

As the table demonstrates, each class is defined by a distinct and highly cohesive vocabulary, validating their separation as unique themes. **Class 1's** lexicon revolves around process governance ('standardization', 'quality'). **Class 2's** vocabulary is dominated by terms of action and automation ('turmalina', 'paste', 'copy', 'generate'), clearly pointing to hands-on testing activities. Finally, **Class**

Table 4: Top 5 Keywords per Thematic Class, Ranked by Chi-squared (χ^2) Value

Class	Top Keywords	χ^2 Value
1 (Red) - Quality & Standardisation (n=18, 34.6%)	standardization	10.45
	end	8.19
	quality	7.11
	type	5.03
	qa	5.03
2 (Green) - Testing & Automation (n=20, 38.5%)	turmalina	15.13
	paste	10.85
	copy	10.85
	tool	9.03
	ai	7.63
3 (Blue) - Specification & BDD (n=14, 26.9%)	write	12.52
	start	11.76
	person	11.11
	project	8.74
	bdd	8.64

3 is characterized by words related to planning and specification ('write', 'start', 'project', 'bdd'). This clear lexical differentiation provides a robust foundation for using these classes as proxies for the main themes discussed by the participants in the subsequent analysis of the research questions.

3.3 Critical STLC phases and day-to-day obstacles (RQ1)

In response to RQ1, data from the focus group voting points conclusively to the initial phases of the STLC. **Requirements Analysis** and **Test Planning** emerged as the main pain points, drawing 21 of the 30 negative votes (11 for requirements and 10 for planning). This perception is reinforced by the polls, where these phases received the highest criticality scores (4.6 and 4.4 out of 5, respectively). Participants cited "ambiguous or shifting requirements" (9 mentions) and "incomplete or outdated test plans" (7 mentions) as the main daily obstacles. One senior tester summarized the group's sentiment by stating that by the time issues reach the execution phase, "we're just counting the scars".

This strong consensus reflects the well-known principle of "shift-left" quality management, where issues in upstream artifacts have an outsized impact downstream. The following tables provide a quantitative breakdown of these practitioner concerns. Table 5 details the perceived criticality scores for each STLC phase, while Table 6 itemizes the top-ranked obstacles and their originating phase in the lifecycle.

Table 5: Perceived criticality of STLC phases ($N = 10$)

STLC phase	Mean score	# negative votes
Requirements Analysis	4.6	11
Test Planning	4.4	10
Test Case Design	3.2	4
Test Execution	3.1	3
Test Closure	2.5	2

The lexical analysis strongly corroborated this emphasis on the challenges of the initial phases. The structure of the DHC, visualized in the dendrogram (Figure 2), shows that two of the three thematic classes—**Quality & Standardisation** and **Specification & BDD**—are grouped together, separate from the automation class, indicating a shared focus on pre-execution activities. The keywords that define these classes, detailed in Table 4, confirm this focus with high-significance terms like `requirement`, `quality`, `standardization`, `write`, and `start`. This indicates that participants see requirements analysis and specification as the primary point of fragility.

Table 6: Top obstacles and their originating STLC phase

Obstacle	Origin phase	# mentions
Ambiguous or shifting requirements	Requirements	9
Incomplete or outdated test plans	Planning	7
Script fragility after UI changes	Execution	4

The convergence between the voting data and the lexical analysis reinforces that practitioners see Requirements Analysis and Test Planning as the make-or-break stages of the testing life-cycle. This confirms that the classic "shift-left" notion—that defects found earlier cost less—still dominates the testers' perception, with ambiguity in requirements and poorly maintained plans identified as primary pain points. The enthusiasm for new technologies like LLMs does not displace this focus; on the contrary, it only highlights the critical need for well-defined inputs for automation to be effective. This finding dovetails with related work in software quality and underlines why our subsequent questions focus on how LLMs might alleviate (or complicate) these specific phases.

Answer to RQ1. Practitioners rank *Requirements Analysis* and *Test Planning* as the most critical—and troublesome—STLC phases, earning 70% of all negative votes. Core blockers are **ambiguous/shifting requirements** and **incomplete or outdated test plans**, which cascade defects and rework downstream. The result echoes long-standing evidence that weak upstream artifacts push testers into downstream "damage-control" rather than prevention.

3.4 Concrete Benefits Attributed to LLMs (RQ2)

Despite some skepticism, participants identified several concrete benefits that LLM-based tools could bring to their workflows. During the affinity-diagram exercise, each participant placed three "+" stickers on statements that captured positive aspects of LLM support, for a total of thirty votes. Open coding of the forty-two positive statements converged on three benefit clusters: *efficiency*, *Coverage*, and *Knowledge acceleration*. Table 7 aligns these clusters with vote counts, code frequencies, and emblematic quotations.

Table 7: Benefit clusters, vote counts and illustrative remarks (*N*=10)

Benefit cluster	"+" votes	Coded refs	Representative quote
Efficiency — automated scaffolding and documentation	12	15	"*Writing boilerplate cases eats half my sprint; an LLM could draft that in minutes.*" — P5
Coverage — edge cases and negative paths	10	12	"*The model surfaces corner inputs we forget when the spec is vague.*" — P3
Knowledge acceleration — upskilling junior testers	8	10	"*Newbies see good assertion patterns instead of just copy-pasting old tests.*" — P9

(a) Efficiency Gains This was the top-voted benefit, garnering 12 of 30 favorable votes. Testers were enthusiastic about offloading tedious, time-consuming tasks to an AI. "Writing boilerplate tests eats half my sprint; an LLM could draft that in minutes", said one participant, envisioning substantial time savings on routine test-case scripting and documentation. This strong focus on operational efficiency was directly mirrored in the lexical analysis. The **'Testing & Automation'** class was the most lexically distinct of the three (see Figure 2), indicating that the most cohesive part of the conversation revolved around hands-on automation. The class's highly significant keywords, such as `turmalina`, `ai`, `generate`, `paste`, and `copy` (see Table 4), provide concrete evidence for the desire to automate manual tasks and reduce repetitive work within their existing toolchains.

(b) Improved Coverage The second major theme, with 10 positive votes, was using LLMs to generate a broader range of test cases, particularly edge cases and negative scenarios. Testers admitted that under tight deadlines, they often focus on "happy paths". An LLM, however, could serve as a creative partner to ensure no obvious category of behavior is forgotten. "The model surfaces corner inputs we forget when the spec is vague," noted one participant, highlighting the LLM's potential to challenge assumptions and bring up cases humans might overlook.

(c) Knowledge Acceleration The third benefit cluster, with 8 votes, revolved around learning and skill transfer. Participants saw LLMs as a way to accelerate the onboarding of junior testers or non-specialists. "Newbies see good assertion patterns instead of just copy-pasting old tests," said one senior tester, suggesting that AI could expose less experienced team members to better practices. This aligns with the idea of an AI as a tutor or pair programmer, helping manual testers learn BDD syntax or providing quick explanations for domain-specific concepts.

Answer to RQ2. Testers foresee three chief benefits from LLM adoption: **(1) Efficiency** – offloading repetitive test scripting and boilerplate generation, freeing time for higher-value tasks. **(2) Coverage** – auto-suggesting edge cases and negative scenarios that humans often miss, broadening fault detection. **(3) Knowledge acceleration** – providing on-demand exemplars that help juniors learn best practices faster. These gains echo recent evidence of AI tools boosting developer productivity and catching extra faults, but their realisation depends on mitigating the limitations outlined in RQ3.

3.5 Limitations and Challenges for LLM Adoption (RQ3)

Turning to the potential downsides, participants were candid about several significant barriers to adopting LLMs in their testing workflow. Three themes dominated the negative affinity votes (24 of 30 "–" votes clustered here): prompt sensitivity, limited domain generalization, and data privacy risk. Table 8 links these themes to vote counts, coded references, and emblematic quotations. "Share" expresses the percentage of the 30 negative ("–") votes that each theme attracted.

(a) Prompt sensitivity: This was the single most cited limitation, attracting 10 negative votes. Testers observed that LLMs can be "unpredictable or misleading if the prompt isn't perfectly crafted". This sentiment is strongly explained by the lexical analysis of the **'Quality & Standardization'** class (see Table 4). The keywords in this class reveal three specific preconditions that testers consider essential for success:

(1) **Well-written requirements:** The prominence of the keyword **`requirement`** directly supports the *"garbage in, garbage out"* sentiment. Testers believe that without clear source requirements, the generated prompts and subsequent tests will be flawed.
(2) **Clear writing standards:** The keyword **`standardization`** reflects the need for clear guidelines and naming conventions. Without them, participants fear that any efficiency gains would be lost to the rework needed to fix inconsistent or non-compliant outputs.
(3) **Rigorous human review:** The keyword **`qa`** highlights the group's skepticism about blindly trusting LLM outputs. It underscores the perceived, non-negotiable need for a final quality assurance check by a human before any generated artifact is accepted.

(b) Limited domain generalisation: The second major concern, with 8 negative votes, was that out-of-the-box LLMs may not understand the specific domain or tech stack of a project. "It handles 'checkout' fine but freezes on PLC and Modbus jargon", noted one participant from an industrial automation project, highlighting the model's struggle with niche terminology. This underscores the need for domain adaptation, either through fine-tuning or retrieval-augmented generation.

(c) Data privacy and security concerns: The third prominent challenge (6 votes) revolves around the sensitivity of code and data. Many organizations have strict NDAs and compliance regulations (like GDPR) that prevent sharing proprietary information with external cloud services. "Our NDA forbids sending production logs outside the network", explained one tester. However, while data privacy was identified as a significant risk in the guided voting exercise, the lexical analysis of the spontaneous discourse offers a nuanced perspective. The analysis found a notable absence of terms related to privacy or data leaks in the corpus. This suggests that while testers acknowledge privacy as an important organizational barrier, their more immediate, top-of-mind operational hurdles are related to the quality of inputs and the standardization of processes.

In summary, RQ3 reveals that practitioners see prompt engineering effort, domain limitations, and privacy risks as the main obstacles to adopting LLMs in testing. These reflect real socio-technical constraints: a tool that is too high-maintenance or too risky won't be embraced, regardless of its raw capabilities. The concerns our participants raised are starting to be acknowledged in emerging research (e.g., calls for better prompt frameworks, domain-specific fine-tuning, and secure AI deployments), but solutions are still in progress. Addressing these will be key to moving from promising prototypes to widespread industry use.

Answer to RQ3. Industrial uptake of LLMs hinges on three hurdles: **(1) Prompt sensitivity** – small wording changes can yield unstable or hallucinated tests; teams will need vetted prompt templates and prompt-engineering know-how. **(2) Domain gaps** – generic models stumble on sector-specific jargon unless augmented (fine-tuning or retrieval of domain context). **(3) Data privacy** – NDAs and regulations bar sending code/logs to public APIs, pressing for on-prem/VPC deployments, or sanitized inputs. Until these issues are addressed—alongside human review—LLM adoption should proceed cautiously.

3.6 Synthesis and Integration of Evidence

This section synthesizes the findings from the various data sources—affinity voting, polling, and the different outputs of the lexical analysis—to form a cohesive interpretation. By triangulating these pieces of evidence, we can identify overarching meta-themes in the practitioners' discourse.

The key insight from this integrated view is the confirmation of a central tension within the testers' perceptions. There is a strong pull towards the benefits of **acceleration** offered by LLM-based automation, but this is counterbalanced by an equally strong understanding that these benefits are entirely contingent on the **standardization** of processes and the quality of upstream artifacts. This "accelerate vs. standardize" dichotomy represents the core socio-technical challenge for adopting LLMs in testing workflows.

Table 8: Salience of the main limitations (N=10) and practitioner mitigation ideas.

Limitation theme	Salience		Representative quote	Mitigation suggested
	"–" votes	Share		
Prompt sensitivity	10	33 %	"*One misplaced adjective and the model invents half the flow.*" — P5	Curated prompt libraries; prompt diff reviews; pairing junior/senior on prompt crafting
Domain generalization	8	27 %	"*It handles 'checkout' fine but freezes on PLC and Modbus jargon.*" — P1	Retrieval-augmented prompts; domain-specific fine-tuning; glossary injection
Data-privacy concerns	6	20 %	"*Our NDA forbids sending production logs outside the network.*" — P4	On-prem or VPC deployment; selective redaction; synthetic-data prompts

4 Final Remarks

The overarching goal of this research was to understand how software testers perceive the benefits, risks, and integration effort of LLMs across the *Software Test Life-Cycle*. To reach that goal, we conducted a focus group with five experienced QA engineers from a Brazilian RD centre. The study combined a semi-structured discussion, Likert polls, and an affinity-diagram voting exercise, followed by a thematic analysis and a lexicometric analysis of the transcribed corpus. The findings from the different data sources converged to answer our three research questions (RQ). For RQ1, both the affinity voting and the lexical analysis pointed to *Requirements Analysis* and *Test Planning* as the most critical STLC phases, with two of the three generated thematic classes focusing on these upstream activities. For RQ2, *Efficiency* was the top-voted benefit, a finding directly corroborated by the distinct *Testing & Automation* class. Finally, for RQ3, while voting highlighted prompt sensitivity, domain gaps, and data privacy, the lexical analysis added an important nuance: the absence of a privacy-related vocabulary suggests that day-to-day operational concerns, such as standardization, are more top-of-mind for this group.

In terms of implications, our study guides both researchers and industry. For researchers, it spotlights underexplored areas like requirement analysis support and reinforces calls for work on prompt engineering and human-AI collaboration. For practitioners, our findings recommend a cautious, phased adoption, starting with LLM assistance in early life-cycle phases, developing prompt libraries, and using AI outputs as a resource for training and onboarding.

As future work, we see several avenues, including the replication of this study in other contexts (with junior testers or in different regions). The limitations of our lexical analysis also point to new research: the 36.4% classification coverage, while indicating stable themes, suggests a lexical heterogeneity that could be explored with different analysis parameters (e.g., a minimum frequency of 2) to potentially capture nuances about privacy risks. Furthermore, the absence of "data governance" vocabulary suggests that this topic should be explicitly explored in future focus group scripts.

In conclusion, the lexical analysis confirms that while the group strongly values the automation offered by LLMs, they consider it viable only upon a solid foundation of requirements and standards. The tension between the "speed" of automation and the "robustness" of processes is the central dilemma identified. This dichotomy should guide both future research and the practical adoption of AI in the STLC, ensuring that the promise of the technology is realized through a careful alignment with human factors and shop-floor realities.

ACKNOWLEDGMENTS

This work has been partially funded by the project 'iSOP Base: Investigação e desenvolvimento de base arquitetural e tecnológica da Intelligent Sensing Operating Platform (iSOP)' supported by CENTRO DE COMPETÊNCIA EMBRAPII VIRTUS EM HARDWARE INTELIGENTE PARA INDÚSTRIA - VIRTUS-CC, with financial resources from the PPI HardwareBR of the MCTI grant number 055/2023, signed with EMBRAPII.

REFERENCES

[1] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, and et al. 2023. Sparks of Artificial General Intelligence: Early Experiments with GPT-4. *arXiv preprint arXiv:2303.12712* (2023). https://arxiv.org/abs/2303.12712
[2] Vahid Garousi, Michael Felderer, and Marco Kuhrmann. 2020. Exploring the Industry's Challenges in Software Testing: An Empirical Study. *Journal of Software: Evolution and Process* 32, 8 (2020), e2251. https://doi.org/10.1002/smr.2251
[3] Fatih Gurcan, Gonca Gokce Menekse Dalveren, Nergiz Ercil Cagiltay, Dumitru Roman, and Ahmet Soylu. 2022. Evolution of software testing strategies and trends: Semantic content analysis of software research corpus of the last 40 years. *IEEE Access* 10 (2022), 106093–106109.
[4] ISTQB. 2024. Certified Tester Foundation Level Syllabus v4.0. https://www.istqb.org. Accessed 4 Jul 2025.
[5] Eriks Klotins, Tony Gorschek, Katarina Sundelin, and Erik Falk. 2022. Towards cost-benefit evaluation for continuous software engineering activities. *Empirical Software Engineering* 27, 6 (2022), 157.
[6] Jyrki Kontio, Laura Lehtola, and Johanna Bragge. 2004. Using the focus group method in software engineering: obtaining practitioner and user experiences. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE'04.* IEEE, 271–280.
[7] Arghavan Moradi Dakhel, Amin Nikanjam, Vahid Majdinasab, Foutse Khomh, and Michel C. Desmarais. 2023. Effective Test Generation Using Pre-trained Large Language Models and Mutation Testing. *arXiv preprint arXiv:2308.16557* (2023). https://arxiv.org/abs/2308.16557
[8] David L. Morgan. 1997. *Focus Groups as Qualitative Research* (2 ed.). SAGE Publications.
[9] Mika V. Mäntylä, Juha Itkonen, and Joonas Iivonen. 2012. Who Tested My Software? Testing as an Organizationally Cross-Cutting Activity. *Software Quality Journal* 20, 1 (2012), 145–172. https://doi.org/10.1007/s11219-011-9157-4
[10] Per Runeson and Martin Höst. 2009. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering* 14, 2 (2009), 131–164. https://doi.org/10.1007/s10664-008-9102-8
[11] Janice Singer, Susan E Sim, and Timothy C Lethbridge. 2008. Software engineering data collection for field studies. In *Guide to advanced empirical software engineering*. Springer, 9–34.
[12] Michele Tufano, Chuning Chen, and Miltiadis Allamanis. 2023. Large Language Models as "Big Assistants" for Test Generation: An Empirical Study. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE '23)*. IEEE, 1234–1245. https://doi.org/10.1109/ASE57332.2023.00098
[13] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2024. Software Testing With Large Language Models: Survey, Landscape, and Vision. *IEEE Trans. Softw. Eng.* 50, 4 (April 2024), 911–936. https://doi.org/10.1109/TSE.2024.3368208
[14] Zhiqiang Yuan, Yiling Lou, Mingwei Liu, Shiji Ding, Kaixin Wang, Yixuan Chen, and Xin Peng. 2024. No More Manual Tests? Evaluating and Improving ChatGPT for Unit Test Generation. *arXiv preprint arXiv:2305.04207* (2024). https://arxiv.org/abs/2305.04207