# When Are Agentic Workflows Worth It? An Exploratory Evaluation of Feedback-to-PRD Generation and Inference Cost

**Aditya Hambire**
*Dept. of Artificial Intelligence and Data Science*
*K. J. Somaiya Institute ofTechnology*
*Mumbai, India*
aditya.hambire@somaiya.edu

***Abstract*—Product requirements documents (PRDs) can be drafted by language models from selected customer feedback, but it is unclear whether adding critique and role structure justifies the extra inference. We report a bounded, exploratory comparison, made after opportunity selection, of three generation procedures: a strong single prompt (A), an iterative single-agent conversation that retains its history (B), and a role/context workflow in which writer, reviewer and editor roles exchange typed artifacts (C). With public issue excerpts and researcher-defined scopes for three products, two allocations and two repeats, 36 slots were planned; 28 produced PRDs and eight stopped at a conservative local reservation ceiling (A 12/12, B 6/12, C 10/12). In the expanded setting, B and C used on average 7,002 and 5,955 more provider-reported tokens per paired slot than A. A retrospective same-model assessment yielded strictly valid records for 11 of 18 unique documents (18 of 28 slots); valid ratings were mostly at the scale ceiling. Two analyses of the evaluation itself follow. A seeded-defect study shows that deterministic evidence checks detect surface defects injected into 18 real PRDs but none of the semantic defects (a wrong-but-valid citation, a negated requirement). A design-level power simulation shows that near-ceiling ratings and assessment attrition leave a 13-case design underpowered for modest effects. The data show neither a reliable quality benefit nor that the additional inference is justified; the contribution is an auditable account of completion, cost and evaluator validity.**

***Index Terms*—requirements engineering, product requirements document, large language models, agentic workflows, customer feedback, LLM-as-a-judge, inference cost**

## I. Introduction

Customer feedback exposes problems but seldom specifies a testable implementation contract. Turning selected feedback into a product requirements document (PRD) requires separating what customers reported from what a team proposes, honouring scope boundaries and attaching observable acceptance criteria [1]. Language models can draft such documents, and orchestration frameworks make it easy to add critique, revision and role-differentiated stages [2], [3]. Each stage consumes inference, and the evidence that it helps is mixed: intrinsic self-correction can fail without an external signal [4], and a single persistent conversation can match homogeneous multi-agent workflows on several benchmarks [5]. We ask whether this carries over to PRD generation, given a strong single-prompt baseline, an iterative baseline and explicit resource accounting.

The experiment is deliberately narrow. Selected feedback and a researcher-defined scope are fixed, and three procedures draft a PRD: a thoroughly specified single prompt (A), one persistent analyst conversation that drafts, critiques and revises (B), and a role/context workflow in which writer, reviewer and editor roles receive typed artifacts instead of the full conversation (C). The comparison starts after opportunity selection and does not measure prioritisation, stakeholder agreement or productivity; C bundles role differentiation with context handling and is not presented as collaboration among independent agents. Three research questions guide the study: **RQ1**, does the role/context workflow improve PRD quality over A and B; **RQ2**, how do differences vary across the product cases; **RQ3**, are measured improvements sufficient to justify the additional inference?

The contributions are bounded by the evidence: (i) a controlled design with shared inputs and criteria and two strong baselines; (ii) complete accounting of all 36 planned slots, separating local reservation stops from provider behaviour; (iii) an assessment protocol whose invalid records are retained, not repaired; (iv) a seeded-defect analysis of what deterministic evidence checks can and cannot detect; and (v) a design-level power analysis. No claim of agent superiority is made.

## II. Related Work

**Feedback to requirements.** Reviews cover language models in requirements engineering [6], [7] and app-review analysis [8]. The closest study, Mallya et al. [9], evaluates five lightweight open models on request classification and specification generation with identical inputs across prompting strategies; specification quality was rated by a single evaluator, as the authors acknowledge. RITA [10] packages these capabilities as a tool. Two 2026 studies address related artifacts, agent-governed retrieval-augmented issue specifications [11] and multi-agent product design specifications [12]; we accessed only their titles and metadata and do not characterise them further. A feedback-to-PRD interface is therefore not itself a contribution.

**Single versus multi-agent workflows.** Role-based frameworks encode software-team procedures as agent roles [2], [3], and debate among model instances has been proposed to improve reasoning [13]. Counter-evidence is growing: debate does not reliably beat self-consistency [14], a single agent with a strong prompt can approach discussion approaches [15], and one agent simulating homogeneous workflows through multi-turn conversation can match them on seven benchmarks [5]. Self-refinement helps some tasks [16], whereas intrinsic self-correction can degrade reasoning [4]. This motivates an iterative single-agent baseline (B) and deterministic external review candidates for both iterative arms.

**Evaluation and cost.** Kapoor et al. argue that agent evaluations should report cost with accuracy [17]; cost-aware model use has been studied directly [18]. LLM judges show position, verbosity and self-enhancement biases [19], can favour their own generations [20], and agree with human evaluators in software engineering only to a task-dependent degree [21]. Claim-level evaluation of generated text against a supplied source follows atomic-fact scoring [22], and seeded-fault injection is a standard way to probe what a detector can find [23]; we use the former to shape the assessment schema and the latter to validate the deterministic checks. Table I summarises the closest studies. The residual gap is narrow: a post-selection PRD task with strong single-prompt and iterative baselines, allocations declared

TABLE I
CLOSEST RELATED STUDIES AND THEIR RELATION TO THIS WORK

| Study | Design and evidence | Relation to this study |
|---|---|---|
| Mallya et al. [9] | Five open models, several prompts; single-evaluator rating | Feedback-to-specification precedent; no matched workflow-cost comparison |
| Xu et al. [5] | Seven benchmarks; one persistent conversation vs. workflows | Motivates B; we make no cache-reuse claim |
| **This study** | PRD from fixed scope; A/B/C; 36 slots accounted; same-model assessment; seeded-defect and power analyses | Exploratory, three cases; no quality or cost-justification claim |

in advance, complete accounting of failed and invalid outcomes, and a validated-record assessment protocol.

## III. PRODUCTSENSE ARCHITECTURE

ProductSense is the application context of the task: a FastAPI backend, a build-free browser frontend, SQLite persistence and LangGraph checkpoints, with live generation through the Gemini Developer API only and a labelled offline demo mode (Fig. 1). Feedback arrives as a bounded CSV (at most 30 rows) or a saved public sample and is treated as untrusted data. Grouping is lexical, ranked by sample share and goal-term coverage (not effort or value), and the user must explicitly approve scope before drafting. The PRD graph performs writing, a model review, at most one revision and a deterministic check that flags structure, citation identifiers and risk patterns as candidates for inspection, not truths. Traceability between requirements and sources is a long-standing concern [24], [25]. A requirement-to-evidence matrix and an evidence ledger were added after the study; they are not part of the experiment, and the matrix's checks are used only in Section VII.

## IV. METHODOLOGY

**Cases.** The unit of analysis is the case (one feature per product); repeats are nested within cases and requirements are not independent observations. Three evaluation cases were selected purposively from public trackers: VS Code find-in-selection (issues #284007, #237774), Joplin note-history restore (#11463) and Nextcloud public-share expiration (#8837); Excalidraw PDF export served only for development. Source excerpts are exact substrings with authors removed. Each case fixes a scope, corrections and three important needs, all researcher-authored instructions rather than stakeholder approvals or observed facts. No source or text overlap between development and evaluation products was found (pretraining exposure is not excluded), and no evaluation output changed prompts, checks or settings.

**Conditions.** All conditions receive the same feedback, scope, corrections, 15 required headings, model alias (`gemini-3.5-flash-lite`), temperature 0.2, schema-constrained JSON output and quality criteria (user-outcome goals; separated evidence and assumptions; no invented numeric targets; cited requirement IDs; observable acceptance criteria including failure or cancellation). Fig. 2 contrasts context handling. A: one call with all criteria and an internal self-check. B: one persistent conversation with three generations (draft, critique, revision) that retains all earlier turns. C: three role calls (writer, reviewer, editor); the reviewer sees the brief, draft and deterministic candidates but not the writer's conversation, and the editor sees the draft and the critique (at most four issues). B receives the same candidates and criteria as C, so the B/C contrast is a bundled difference in role framing and context isolation. Seeds were $20261025 + r$ for repeat $r$, identical across conditions; seeded sampling is not a determinism guarantee.

**Allocations and reservation.** Two allocations were fixed before evaluation: compact (output caps A 4,000; B and C 1,400+600+2,000; ceiling $L_b = 40{,}000$) and expanded (A 6,000; B and C 2,200+800+3,000; $L_b = 60{,}000$), with 3 cases × 3 conditions × 2 allocations × 2 repeats = 36 slots. Equal output caps do not imply equal realised inference. Before each call the runner computes a conservative proxy: with $\hat{h}_k$ the UTF-8 byte length of system text, messages and schema plus a 1,024-byte margin, $o_k$ the stage output cap and $L_b$ the cumulative ceiling, stage $k$ is admitted only if

$$\sum_{j=1}^{k} \hat{h}_j + o_j \le L_b, \tag{1}$$

otherwise the slot ends with a local `token_budget` stop and no call is made. Because $\hat{h}_k$ grows with a retained conversation, Eq. (1) is a byte-based proxy, not a tokenizer count, and is distinct from provider-reported usage.

**Structural checks.** Every final PRD was checked for all 15 headings and for linked Given/When/Then criteria on every numbered requirement. The checker also emits lexical flags (an unknown evidence identifier, a requirement without a valid identifier or provenance label, vague acceptance language); these are candidates for inspection, not verdicts.

**Retrospective model assessment.** No human rating was completed. An assessment protocol was frozen before any assessor request. The same model alias (temperature 0.2, seed 20261026) assessed each unique packet (context, scope, corrections, feedback, three needs and PRD) without condition labels, resource data or critiques, under neutral content-derived identifiers in shuffled order; slots with identical text share one assessment, and two predefined packets received one repeat (20 calls in total). The coordinating researcher had seen condition identities, so masking is procedural. The assessor returns the status of each need, inventories of requirements and atomic factual claims with support status (aimed at unsupported claims of the kind surveyed as hallucination [26]), 1–5 ratings (5 = no identified gap) for scope, corrections, provenance and clarity, and findings with literal excerpts. Scripts check that every excerpt occurs literally in the PRD or source and that identifiers, categories and ranges are valid; literal presence is not entailment. An invalid record is retained unchanged, excluded from primary aggregates, never rescored and not replaced by a repeat. No composite accuracy score is defined.

**Analysis.** Missing PRDs are execution outcomes without a quality value; invalid assessments and unknown token usage are unavailable, not zero. Contrasts are paired by case, allocation and repeat, averaged over repeats within each case, then over cases. For a measure $m$ (reported tokens or covered needs) and conditions $X, Y$,

$$\Delta_m^{(X-Y)} = \frac{1}{|P_b|} \sum_{p \in P_b} \frac{1}{|R_p|} \sum_{r \in R_p} m_{p,r}^{X} - m_{p,r}^{Y}, \tag{2}$$

where $P_b$ is the set of cases with a usable pair in allocation $b$ and $R_p \subseteq \{1, 2\}$ the repeats with both members usable. For tokens in the expanded allocation all six pairs are usable. Only descriptive means and ranges are reported; with three purposive cases no intervals or significance tests are computed.

## V. EXPERIMENTAL SETUP

Every generation attempt was reserved in a persistent ledger before the network call; request, parsed output, raw response, finish

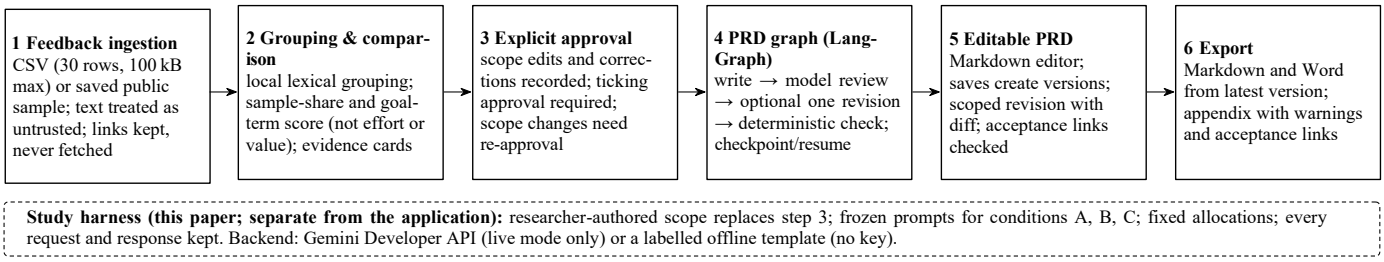


Fig. 1. ProductSense architecture. Solid arrows: interactive workflow. The dashed study harness reuses the provider layer, store and deterministic checker but supplies the approved scope itself and applies the frozen experimental prompts; it is not the interactive approval path.

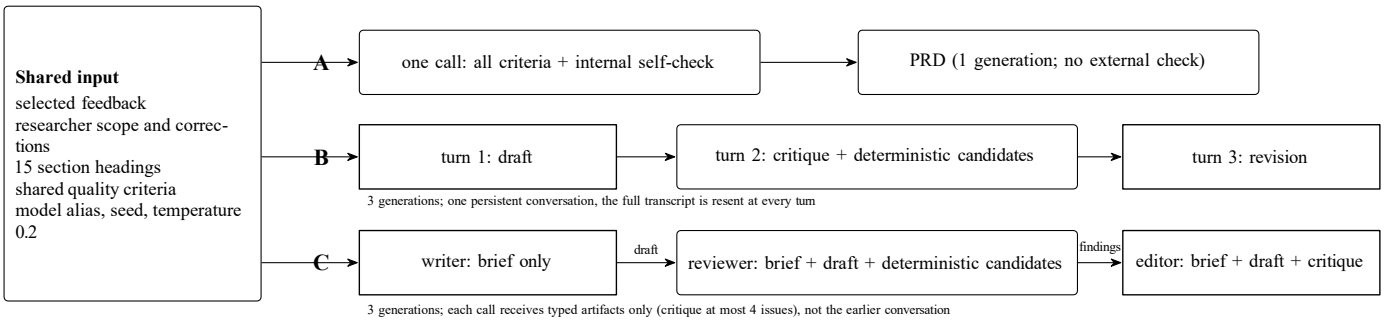


Fig. 2. The three generation procedures and their context handling. A uses one call; B resends a growing history at every turn; C passes only typed artifacts between role-specific calls. B and C share review criteria and deterministic candidates.

TABLE II
COMPLETION AND PROVIDER-REPORTED TOKENS BY ALLOCATION

| Alloc. | Cond. | PRDs | Calls | Prompt | Output | Total |
|---|---|---|---|---|---|---|
| compact | A | 6/6 | 6 | 8,556 | 5,835 | 14,391 |
| compact | B | 0/6 | 12 | 25,071 | 6,398 | 31,469 |
| compact | C | 4/6 | 16 | 33,720 | 8,574 | 42,294 |
| expanded | A | 6/6 | 6 | 8,556 | 5,835 | 14,391 |
| expanded | B | 6/6 | 18 | 43,969 | 12,432 | 56,401 |
| expanded | C | 6/6 | 19 | 39,580 | 10,543 | 50,123 |
| Evaluation total | | 28/36 | 77 | 159,452 | 49,617 | 209,069 |

Calls include one failed request (expanded C) with unknown usage, so that row is a lower bound. With development: 87 attempts, 86 responses, 239,303 known tokens.

TABLE III
PAIRED DIFFERENCES, EXPANDED ALLOCATION (MEAN OVER CASES)

| | Reported tokens | | | | Needs |
|---|---|---|---|---|---|
| Contrast | Total | Prompt | Output | Pair range | covered |
| B−A | +7,001.67 | +5,902.17 | +1,099.50 | 6,170 to 7,740 | +0.25 |
| C−A | +5,955.33 | +5,170.67 | +784.67 | 5,406 to 6,922 | +0.25 |
| C−B | −1,046.33 | −731.50 | −314.83 | −1,880 to −464 | 0.00 |

Token columns: six pairs. Coverage uses valid pairs only (Joplin two repeats, Nextcloud one, VS Code none); descriptive.

status, returned model identity, provider usage counters and stage time were saved. Interrupted attempts counted as spent; generation stopped on provider failure, truncation or model-identity change, with no automatic retry or paid fallback. Code, prompts and data were frozen before development and again before evaluation; eight chained amendments record operational and reporting changes, none altering prompts, criteria, data or model. The cumulative ceiling of 180 attempts applied throughout, and no output was polished or rerun to obtain better results.

Development used 10 calls and 30,234 reported tokens; evaluation used 77 attempts. One request in the Nextcloud expanded C slot returned a provider-unavailable error with unknown usage, and a prospectively authorised one-use recovery succeeded; this is not evidence of quota exhaustion. The ledger holds 107 attempts (87 generation, 20 assessment; 86 generation responses succeeded). Usage is taken from provider counters [27] (thought counters were zero in all generation records), with generation and assessment usage kept separate. Latency is not analysed; no cache reuse or monetary cost is claimed.

## VI. RESULTS

**Completion and resources.** Table II and Fig. 3 report all 36 slots. A completed 12/12, B 6/12 and C 10/12. All six compact B slots stopped after the critique call because the retained conversation pushed the revision reservation beyond the 40,000-unit ceiling, and both compact VS Code C slots stopped likewise; all 18 expanded slots completed. These are local policy outcomes, not provider truncations, and say nothing about the missing PRDs' quality. Compact and expanded A rows (and the four completed compact C slots) coincide because the texts and counters are identical.

In the expanded allocation, B and C used far more reported tokens than A. Using Eq. (2) over six case/repeat pairs, B used on average 7,001.67 more tokens than A, C 5,955.33 more than A, and C 1,046.33 fewer than B (Table III, Fig. 4a). About 84–87% of the extra tokens are prompt tokens, consistent with A sending the inputs once, B resending its history and C passing artifacts rather than the conversation (the design does not isolate this). One pair (Nextcloud repeat 1) contains the failed request, so its C total is a lower bound: the C−A difference there is understated and the C−B difference could be less negative than recorded.

**Structural outcomes.** All 28 PRDs contain the 15 headings and every numbered requirement has linked acceptance criteria, so these

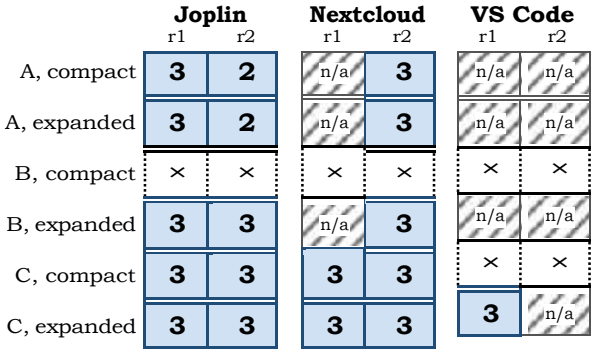


Fig. 3. Status of all 36 slots. Numbers: needs covered (of three) for slots with a valid primary assessment; hatched: PRD without a valid assessment (n/a); dotted ×: local reservation stop, no PRD.

TABLE IV
VALID MODEL ASSESSMENTS AND MISSINGNESS BY PRODUCT AND CONDITION

| Product | Cond. | PRDs | Valid (unique) | Needs | Clarity |
|---|---|---|---|---|---|
| Joplin | A, B, C | 4, 2, 4 | 4(2), 2(2), 4(2) | 2.5, 3, 3 | 4.5, 5, 5 |
| Nextcloud | A, B, C | 4, 2, 4 | 2(1), 1(1), 4(2) | 3, 3, 3 | 5, 5, 5 |
| VS Code | A, B, C | 4, 2, 2 | 0(0), 0(0), 1(1) | n/a, n/a, 3 | n/a, n/a, 5 |

Means over valid slots only (needs of 3; clarity 1–5): descriptive model ratings, not human judgements or an accuracy percentage. PRDs = slots with a final document.

checks do not discriminate between conditions. The checker's error-level lexical flags (unknown evidence identifier, requirement without a valid identifier or provenance label, vague acceptance language) were raised in 8 of 12 A documents (18 flags, 1.50 per document; 16 flags and 1.33 if the two vague-language flags are excluded), 1 of 6 B (0.17) and none of 10 C. All 16 unknown-identifier flags arise in A documents and cite non-feedback inputs, not invented feedback identifiers: the identifier of the selected opportunity (`SELECTED-1`, six times) and the section markers "corrections" (six), "goals" (two) and "Evidence-informed" (two). Since B and C receive these candidates as input and A does not, this confounds workflow with checker access and is not evidence that C is more accurate.

**Assessment attrition.** The 28 PRDs correspond to 18 unique packets (ten shared by two slots, eight single). Eleven primary records, representing 18 slots, passed strict validation; seven, representing ten slots, failed (six with a nonliteral excerpt, one with a duplicated factual excerpt). Table IV shows that validity is uneven: 10 slots for Joplin, 7 for Nextcloud and 1 for VS Code, so no product ranking is supportable. A later audit reproduced every stored validation error: of the seven invalid records, four contain an excerpt that cannot be located even after whitespace and Markdown normalisation, and three fail only through formatting differences or a repeated quotation. No record was changed; the strict rule remains primary.

**Validated semantic results.** Of the 18 valid slots, 16 receive the maximum on all four ratings and cover all three needs. The other two are copies of one Joplin A document (repeat 2), rated 4 throughout and covering two needs, the third (reporting a missing or unreadable revision) being marked missing. Every valid fact inventory held one or two atomic claims, none unsupported; inventories this small are not evidence of freedom from hallucination. Paired contrasts rest on two cases; the mean differences are 0.25 needs for B−A and C−A and 0 for C−B (Table III), arising entirely from the one Joplin repeat in which A scored lower. Because compact and expanded A and C texts coincide, the compact C−A contrast (also 0.25) reuses the same three document pairs. Of two predefined stability pairs, one (Joplin C) had valid records with unchanged ratings; the other (Nextcloud A) had an invalid primary record, so one pair cannot estimate assessor reliability.

TABLE V
SEEDED-DEFECT DETECTION: REQUIREMENTS WHERE THE SPECIFIC FLAG NEWLY APPEARED

| Injected defect | Sites | Final | Pre-amend. (d/n) | App check[a] |
|---|---|---|---|---|
| invented number | 62 | 62/62 | 62/62 | 62 |
| criteria dropped | 58 | 58/58 | 58/58 | 58 |
| G/W/T removed | 62 | 62/62 | 62/62 | 62 |
| unattributed current claim | 62 | 62/62 | 50/62 | 1 |
| vague wording | 62 | 62/62 | 62/62 | 62 |
| label removed | 24 | 24/24 | 23/24 | 23 |
| citations removed | 30 | 30/30 | 29/32 | 29 |
| unknown source ID | 30 | 30/30 | 30/30 | 30 |
| valid citation swapped* | 16 | 0/16 | 0/16 | 0 |
| requirement negated* | 61 | – | – | 1 |
| No-op control (false alarms) | 62 | 0 | 0 | 0 |

"Final" is the reported run (denominator = sites); "Pre-amend." is the run before the two checker amendments below (different eligible sites, hence its own denominator). *Semantic defect: no designated detector exists (–).
[a]Original application check: sites where *any* new issue appeared (an upper bound, not specific).

## VII. VALIDITY OF THE DETERMINISTIC CHECKS: A SEEDED-DEFECT ANALYSIS

Because every PRD passed the structural checks and the model ratings were at ceiling, the instruments themselves need evidence. We asked what the deterministic evidence checks can detect, using known defects injected into the 18 unique evaluation PRDs (62 numbered requirements, three products) and the post-study requirement-to-evidence checker. The protocol, with ten injected defects, the expected detector for each and the defects predicted to go undetected, was written before the first run. Each defect is injected into one requirement chosen by a seeded draw (up to five per PRD per defect) and counts as detected if the specific flag newly appears on it; a no-op edit serves as a false-alarm control. No model is involved.

The no-op control produced 0 false alarms in 62 sites. Surface-pattern defects (an invented numeric target, dropped or malformed acceptance criteria, vague wording, removed provenance, an unknown source identifier, an unattributed current-status claim) were detected at every eligible site; this is largely by construction, because the injected wording matches the detector's patterns, so it is a regression test, not a recall estimate. The informative rows are the semantic defects: replacing a citation by a different valid one was detected at 0 of 16 sites and negating a requirement at 0 of 61 (the original application check flagged one negation site, coincidentally). Lexical checks therefore cannot see wrong-but-valid citations or contradictions; that gap must be filled by model and human assessment.

The analysis also shows how an instrument can mislead. The first run used only 7 of 18 PRDs because the harness located requirement lines only when bulleted (a coverage bias, since corrected). Error analysis of the second run exposed two checker weaknesses, ordinary prose such as "proposed" counting as a provenance label and an attribution word anywhere on a line suppressing a current-status flag; both were amended after the results were seen. Before amendment the

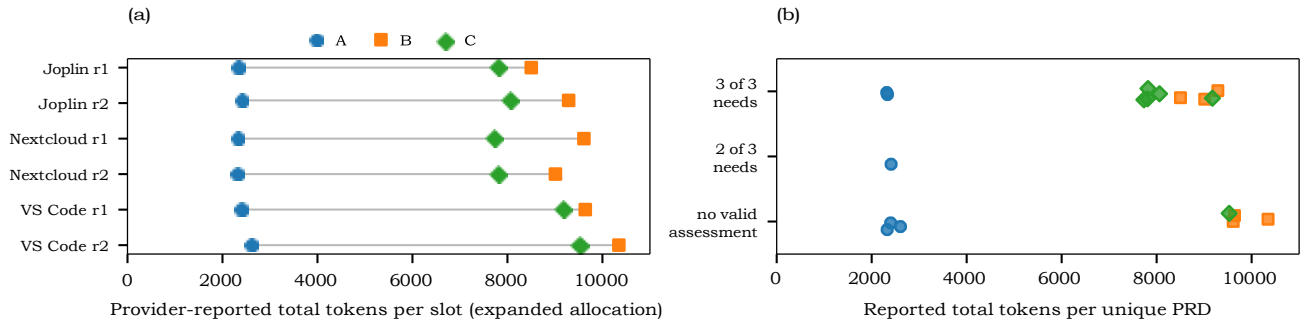


Fig. 4. (a) Provider-reported total tokens per slot, expanded allocation, by case and repeat. (b) The same measure for the 18 unique PRDs grouped by assessed need coverage; "no valid assessment" marks documents whose primary record failed validation. Markers are documents, not independent samples.

same defects were detected at 50/62 (unattributed current claim), 29/32 (citations removed) and 23/24 (label removed) sites, so the amended checker is not held out from these PRDs; the unmutated PRDs raised 8 unknown-identifier, 3 weak-lexical-link and 1 no-provenance flags before and after, so the amendment did not simply add false alarms. All runs are retained.

## VIII. Discussion

**RQ1.** The evidence does not show that the role/context workflow improves PRD quality. Where valid pairs exist, B and C each exceed A by 0.25 needs and equal each other, but this arises from one repeat in one case, under a ceiling, with 10 of 28 slots unassessable; it gives no basis to prefer role separation over an instructed iterative analyst.

**RQ2.** Behaviour varied by case. Compact C completed for Joplin and Nextcloud but not VS Code, which also had the highest expanded token use (mean totals for A/B/C: 2,503/9,993/9,352 for VS Code versus 2,371/8,896/7,940 for Joplin), and validated documents ranged from all six unique Joplin PRDs to one of six for VS Code. Product, feature and feedback condition are confounded, so no robustness claim follows.

**RQ3.** The extra inference is measurable: B and C used on average 7,002 and 5,955 more reported tokens per slot than A, with three calls versus one. Evidence that this buys a worthwhile quality gain is insufficient: valid differences are small, concentrated and ceiling-bound, and no utility function, monetary valuation or editing-effort measure exists. We therefore establish no quality–cost threshold.

**Design-level power.** To judge what a larger follow-up could establish, we simulated the planned case-level paired comparison. The simulation is assumption-driven, not derived from our data: each PRD has three needs; coverage follows a logistic model with a case-difficulty random effect shared by the two conditions; assessments are lost independently; and the test is an exact (Monte Carlo above 12 non-zero differences) sign-flip test on case-level differences at $\alpha = 0.05$. It controls false positives (rejection at most 0.04 with no effect). With coverage near the ceiling (baseline 0.9, as most valid historical assessments were) and 30% assessment loss, 13 cases give power of only 0.11 even for a large effect (log-odds shift 1.5, an odds ratio of about 4.5); 26 cases reach 0.60. At baseline 0.7 without loss, 13 cases give 0.47 at shift 1.0 and 0.77 at 1.5. A follow-up on this scale is therefore descriptive, not confirmatory, unless the outcome has headroom and attrition is low.

**Implications.** First, a reservation policy can dominate a workflow comparison: under the compact ceiling the growth of a retained conversation, not a measured quality difference, prevented every B revision, so completion denominators and proxy versus reported usage should be reported, in the spirit of cost-aware agent evaluation [17]. Second, structural conformance is a poor discriminator, since all 28 documents passed. Third, validation attrition is itself a result: strict literal-excerpt rules rejected records that may hold useful judgements, while quote-level validation can still support wrong inferences. Fourth, evaluation instruments need their own validation: deterministic checks miss semantic defects and are no longer held out once amended after seeing results. Fifth, the deterministic candidates given to B and C were not isolated, so their effect is unknown, and caution about self-correction without external signals [4] applies. A follow-up should test tasks where a strong single prompt visibly misses requirements, with an independent reviewer, calibrated actual-token budgets and measured editing effort; selective escalation after a failed check [18] was not tested.

## IX. Threats to Validity

We use the usual four categories [28]. **Internal.** B and C differ in role framing and context handling together; B and C, unlike A, receive deterministic candidates; seeded sampling is not deterministic; the alias may drift despite identity checks; execution spanned several epochs; and analysts saw conditions before the retrospective assessment. Analysing only valid assessments is selective and not missing at random. **External.** Three purposive cases with one feature each, two repeats, historical public excerpts and one model family limit generality; public data may appear in pretraining. **Construct.** Scopes and needs are researcher-defined; ratings are ceiling-prone; checks, excerpt validation and model ratings are proxies for correctness and value; same-model assessment risks self-preference and correlated errors [19], [20], and agreement with human evaluators is task-dependent [21]; no human rating exists. The seeded defects are synthetic, their detection is partly circular, the checker was amended after results were seen, and the power analysis rests on assumed parameters. **Conclusion.** With about three unique valid document pairs and ceiling ratings the effective information is small, so only descriptive values are reported.

## X. Conclusion and Future Work

In this exploratory study, the iterative and role/context procedures consumed clearly more provider-reported tokens than a strong single prompt, and a conservative reservation policy decided which iterative slots finished. The retrospective assessment offers limited observations under heavy validation attrition, and the seeded-defect analysis shows that the deterministic checks cannot see semantic defects, so the evidence supports neither a claim that agentic variants are better nor that their added inference pays off. It contributes an auditable

design, complete outcome accounting, visible assessment failures and a measured account of what the evaluation instruments can detect. Next steps are blinded multi-rater human assessment, budgets based on actual reported tokens, more products, features and repeats, ablations separating role framing from context isolation, and studies of editing effort. A follow-up design has been drafted, neither frozen nor executed, and no results from it are claimed: 22 product-feature cases (13 in an untouched final split from four products disjoint from development and validation) built from 38 public issue excerpts with provenance records, budgets on actual reported tokens, a typed claim-level assessment schema and a blinded human-review package.

## Data Availability

**Artifacts.** The frozen study package (manifests, protocols, amendments, raw requests and responses with usage, assessment packets and all assessment records including invalid ones, validators and analysis scripts) exists as a source snapshot; all 717 recorded SHA-256 digests match. A script recomputes the quantitative claims of Section VI and Tables II–IV from the raw records, and the tables and figures of this paper are generated from those records by script; one wording error (the flag definition) and one rounding convention found by that audit were corrected. Credentials and runtime databases are excluded; the package is not publicly released and no public deployment exists. Regenerating PRDs requires private credentials and may not reproduce identical text.

## References


[1] ISO/IEC/IEEE, "Systems and software engineering—life cycle processes—requirements engineering," ISO/IEC/IEEE 29148:2018, 2nd ed., Nov. 2018.

[2] S. Hong *et al.*, "MetaGPT: Meta programming for a multi-agent collaborative framework," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.

[3] C. Qian *et al.*, "ChatDev: Communicative agents for software development," in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics (Vol. 1: Long Papers)*, 2024, pp. 15174–15186.

[4] J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, and D. Zhou, "Large language models cannot self-correct reasoning yet," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.

[5] J. Xu, A. Koesdwiady, S. Bei, Y. Han, B. Huang, D. Wang, Y. Chen, Z. Wang, P. Wang, P. Li, and Y. Ding, "Rethinking the value of multi-agent workflow: A strong single agent baseline," arXiv:2601.12307 (preprint), 2026.

[6] M. A. Zadenoori, J. Da˛browski, W. Alhoshan, L. Zhao, and A. Ferrari, "Large language models (LLMs) for requirements engineering (RE): A systematic literature review," arXiv:2509.11446 (preprint), 2025.

[7] C. Arora, J. Grundy, and M. Abdelrazek, "Advancing requirements engineering through generative AI: Assessing the role of LLMs," in *Generative AI for Effective Software Development*, A. Nguyen-Duc, P. Abrahamsson, and F. Khomh, Eds. Cham, Switzerland: Springer, 2024, pp. 129–148.

[8] J. Da˛browski, E. Letier, A. Perini, and A. Susi, "Analysing app reviews for software engineering: a systematic literature review," *Empirical Software Engineering*, vol. 27, no. 2, 2022, art. no. 43, doi: 10.1007/s10664-021-10065-7.

[9] M. A. Mallya, A. Ferrari, M. A. Zadenoori, and J. Da˛browski, "From online user feedback to requirements: Evaluating large language models for classification and specification tasks," arXiv:2510.23055 (preprint), 2025.

[10] M. A. Mallya, A. Ferrari, M. A. Zadenoori, and J. Da˛browski, "RITA: A tool for automated requirements classification and specification from online user feedback," arXiv:2601.11362 (preprint), 2026.

[11] Z. Wang, J. Li, X. Guo, and T. Tsuchiya, "From noisy feedback to evidence-aware issue specifications: an agent-governed retrieval-augmented generation approach," *Automated Software Engineering*, 2026, doi: 10.1007/s10515-026-00640-x.

[12] P. Jiang and J. Han, "Multi-agent systems improve structural quality of AI-generated sustainable product design specifications," *Research in Engineering Design*, 2026, doi: 10.1007/s00163-026-00506-z.

[13] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving factuality and reasoning in language models through multiagent debate," in *Proc. 41st Int. Conf. Mach. Learn. (ICML)*, PMLR vol. 235, 2024, pp. 11733–11763.

[14] A. P. Smit, N. Grinsztajn, P. Duckworth, T. D. Barrett, and A. Pretorius, "Should we be going MAD? A look at multi-agent debate strategies for LLMs," in *Proc. 41st Int. Conf. Mach. Learn. (ICML)*, PMLR vol. 235, 2024, pp. 45883–45905.

[15] Q. Wang, Z. Wang, Y. Su, H. Tong, and Y. Song, "Rethinking the bounds of LLM reasoning: Are multi-agent discussions the key?" in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics (Vol. 1: Long Papers)*, 2024, pp. 6106–6131.

[16] A. Madaan, N. Tandon, P. Gupta *et al.*, "Self-refine: Iterative refinement with self-feedback," in *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

[17] S. Kapoor, B. Stroebl, Z. S. Siegel, N. Nadgir, and A. Narayanan, "AI agents that matter," *Trans. Mach. Learn. Res.*, 2025.

[18] L. Chen, M. Zaharia, and J. Zou, "FrugalGPT: How to use large language models while reducing cost and improving performance," *Trans. Mach. Learn. Res.*, 2024.

[19] L. Zheng, W.-L. Chiang, Y. Sheng *et al.*, "Judging LLM-as-a-judge with MT-Bench and chatbot arena," in *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

[20] A. Panickssery, S. R. Bowman, and S. Feng, "LLM evaluators recognize and favor their own generations," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[21] R. Wang, J. Guo, C. Gao, G. Fan, C. Y. Chong, and X. Xia, "Can LLMs replace human evaluators? An empirical study of LLM-as-a-judge in software engineering," *Proc. ACM Softw. Eng.*, vol. 2, no. ISSTA, Jul. 2025, art. no. ISSTA086, doi: 10.1145/3728963.

[22] S. Min *et al.*, "FActScore: Fine-grained atomic evaluation of factual precision in long form text generation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2023, pp. 12076–12100.

[23] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Trans. Softw. Eng.*, vol. 37, no. 5, pp. 649–678, Sep./Oct. 2011, doi: 10.1109/TSE.2010.62.

[24] O. C. Z. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," in *Proc. 1st Int. Conf. Requirements Eng. (ICRE)*. IEEE, 1994, pp. 94–101.

[25] J. Cleland-Huang, O. C. Z. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman, "Software traceability: Trends and future directions," in *Proc. Future of Software Eng. (FOSE)*, 2014, pp. 55–69.

[26] Z. Ji *et al.*, "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, 2023, doi: 10.1145/3571730.

[27] Google, "Understand and count tokens," Gemini API docs, 2026. [Online]. Available: https://ai.google.dev/gemini-api/docs/tokens

[28] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer, 2012, doi: 10.1007/978-3-642-29044-2.