Context Dependency in LLM-Based Requirements Clarification: Emerging Results from a Controlled SDD Study

Lucas Santos Andrade
Federal University of Campina Grande (UFCG)
Campina Grande, Brazil
lucas.andrade@virtus-cc.ufcg.edu.br

Izabella Ribeiro de Souza Silva
Federal University of Campina Grande (UFCG)
Campina Grande, Brazil
Izabella.silva@copin.ufcg.edu.br

Emanuel Dantas Filho
Federal Institute of Pernambuco (IFPE)
Jaboatão dos Guararapes, Brazil
emanuel.filho@jaboatao.ifpe.edu.br

Ademar Sousa Neto
Federal University of Campina Grande (UFCG)
Campina Grande, Brazil
ademar.1409131@educar.rn.gov.br

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

Danyllo Albuquerque
Federal Institute of Paraíba (IFPB)
Campina Grande, Brazil
danyllo.albuquerque@ifpb.edu.br

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

Abstract

Specification-Driven Development (SDD) workflows increasingly use Large Language Model (LLM)-based agents to analyze natural-language requirements and generate clarification questions. However, User Stories are intentionally concise, and relevant information may be distributed across glossaries, business rules, prior decisions, and system documentation. This paper reports emerging results from a controlled exploratory study on how different context types affect LLM-based requirements clarification. We analyze four User Stories from different domains under six conditions: no additional context, lexical (terms and definitions), operational (rules and exceptions), decisional (prior decisions), systemic (actors and dependencies), and all contexts combined. The 24 executions produced 118 primary questions, which were segmented into 42 normalized informational gaps and classified by context category, coverage, relevance, and behavior across conditions. Operational and combined-context conditions were associated with fewer observed necessary-and-unanswered gaps, while richer context may also surface clarification needs not observed without additional context. The study contributes an evaluation structure for analyzing context dependency in SDD clarification workflows through normalized informational gaps, coverage, relevance, and behavior across conditions.

Keywords

Large Language Models, Requirements Engineering, Requirements Clarification, Specification-Driven Development, User Stories

1 Introduction

User Stories are widely used in agile software development to express requirements in a concise and stakeholder-oriented format. Their compact structure supports communication, prioritization, and iterative refinement, but also makes them inherently dependent on further conversation and contextual interpretation. Prior work has shown that User Stories frequently exhibit quality problems such as ambiguity, incompleteness, underspecification, and lack of clarity, motivating frameworks and tools for assessing and improving their quality [1, 4, 7, 8].

The increasing adoption of Large Language Models (LLMs) in Requirements Engineering has created new opportunities for automating or supporting activities such as requirements analysis, clarification, validation, classification, and specification generation [2, 6, 9]. In this context, Specification-Driven Development (SDD) workflows have emerged as a way to use natural-language specifications as central artifacts for guiding downstream planning, implementation, and validation activities. Tools such as Spec Kit operationalize this idea by providing structured workflows in which LLM-based agents analyze requirements, identify missing information, and generate clarification questions before producing subsequent development artifacts [5].

However, requirements clarification in SDD workflows introduces a specific context-dependency problem. A User Story alone may not contain all information needed to define the expected behavior, but relevant information may already exist in other project artifacts, such as glossaries, business rules, previous decisions, related requirements, interfaces, and system documentation. If an LLM-based agent receives only the User Story, it may ask legitimate clarification questions about missing information. If it receives additional context, some of these questions may disappear, but other behaviors may also occur: the agent may continue asking about information that was already provided, or the added context may reveal new dependencies and generate additional clarification needs. This possibility has received little attention in previous work, which often treats additional context primarily as a mechanism for reducing ambiguity or eliminating missing information. However, richer context may also expose previously hidden dependencies, constraints, or exceptions, leading to new legitimate clarification needs.

Although recent studies discuss LLMs in Requirements Engineering [2, 6, 9], existing LLM-for-RE studies provide limited empirical evidence on how distinct categories of supplied project context change the clarification gaps surfaced by an SDD agent.

The main contribution of this paper is methodological: an evaluation structure based on normalized informational gaps, informational coverage, functional relevance, and cross-condition behavior. Although instantiated here with Spec Kit and GPT-5.5, these analytical dimensions can be reused to evaluate other context-sensitive LLM-based requirements clarification workflows.

The study is guided by the following research questions:

RQ1. How does contextual information affect the number of necessary and unanswered clarification gaps produced by an SDD agent?

Table 1: User Stories used in the exploratory study.

ID	Domain	Main functionality
US02	Waste management and recycling	Search for nearby facilities
US08	Agile software estimation	Simultaneous estimate reveal
US19	Training management	Join a class waitlist
US22	Content recommendation	Recommend content from similar profiles

RQ2. To what extent are gaps associated with each informational function no longer observed when lexical, operational, decisional, or systemic context is supplied?

RQ3. What patterns of persistence, redundancy, and emergence are observed after contextual information is introduced?

2 Study Design

This study is a preliminary execution of a broader protocol for evaluating context dependency in LLM-based requirements clarification. The full protocol foresees additional User Stories and repeated executions, but this emerging-results paper focuses on four User Stories and one execution per condition due to the high annotation effort observed during data collection.

2.1 Cases and Context Conditions

The study used four User Stories selected from Dalpiaz's *Requirements data sets (user stories)* collection [3], which contains stories obtained from public sources and software companies. The cases were intentionally selected from different domains to provide functional diversity rather than statistical representativeness. Table 1 summarizes them.

For each User Story, we manually constructed four three-unit context blocks: lexical (terms and domain expressions), operational (rules, limits, and exceptions), decisional (prior decisions and rationale), and systemic (actors, dependencies, interfaces, and related requirements). Each unit was a self-contained contextual statement, and all blocks were constructed before clarification execution according to the predefined informational categories. CT combined all four blocks. Complete materials are available in the supplementary repository.

2.2 Execution Procedure

The clarification workflow was executed using Spec Kit [5], specifically the `/speckit.clarify` command. According to the protocol, the experimental environment was configured with `specify 0.12.8.dev0` (Spec Kit), `codex-cli 0.143.0`, and `GPT-5.5` as the underlying model. The clarification workflow allows at most five primary questions per execution; therefore, each run represents the agent's prioritized clarification needs rather than an exhaustive set. Each execution was performed in a clean session and under an audit-only configuration that restricted the agent to the supplied User Story and contextual materials [5]. One execution was performed for each User Story–condition pair:

$$4 \text{ User Stories} \times 6 \text{ conditions} = 24 \text{ executions}. \quad (1)$$

The initial protocol planned repeated executions to analyze output stability, but this proved impractical in the preliminary study because the volume of generated questions created a heavy manual analysis burden. Across the 24 executions, 177 clarification-question records were initially extracted from the interaction logs. Of these, 59 duplicates were retained in the source dataset for traceability but excluded from the primary-question set and metric computation, leaving 118 primary clarification questions.

Because a single clarification question may express one or more informational needs, and semantically equivalent needs may appear in different textual forms, the primary clarification questions were not treated as the final unit of analysis.

The workflow comprised four steps: context preparation, clarification execution, question extraction, and segmentation and classification into normalized informational gaps.

2.3 Unit of Analysis, Normalization, and Annotation Procedure

The main unit of analysis in this study is the *normalized informational gap*, rather than the literal clarification question. This choice is important because the same informational need may be expressed through different textual formulations, and a single question may contain multiple independent gaps. For instance, "What does a similar profile mean?" and "How is profile similarity defined?" may both be mapped to the normalized gap Definition of profile similarity, because they request the same underlying definition.

Segmentation and normalization followed a study-specific manual procedure. First, each primary clarification question was inspected to identify whether it expressed one or multiple independently answerable informational needs; compound questions were segmented accordingly. For example, a question asking who can approve a request and under which conditions would yield *Authorized approver* and *Approval conditions*. Each resulting unit was then represented by a concise canonical label describing the information being requested. Units from the same User Story were mapped to the same normalized gap when they requested the same underlying information and could be resolved by the same expected answer or contextual evidence. They were kept separate when they required distinct answers, such as different actors, rules, conditions, decisions, data elements, or exceptions.

After normalization, each gap is classified along three complementary dimensions:

- **Contextual category**: lexical, operational, decisional, systemic, or unassociated;
- **Informational coverage**: unanswered, partially answered, or already answered/redundant; and
- **Functional relevance**: necessary (blocking), useful (non-blocking), or non-pertinent.

A gap was classified as *necessary (blocking)* when resolving it was required to determine or implement the core functional behavior without an unsupported assumption; it was classified as *useful (non-blocking)* when it could refine the requirement but was not required for the core functionality. Out-of-scope gaps were classified as *non-pertinent*.

The same informational gap can surface independently across multiple conditions for a given User Story. A gap first identified in C0, for example, may reappear unchanged once context is introduced. To account for this, per-condition gap counts were deduplicated across all six conditions within each User Story. This cross-condition deduplication reduced the 115 per-condition occurrences summed in Table 2 (18+20+18+20+19+20) to 42 unique normalized gaps, which serve as the unit of analysis for the category-specific elimination results in Section 3.

In addition, after normalization, gaps observed in the baseline condition C0 are compared with those observed in the contextual conditions. This comparison allows each gap to be classified as:

- **eliminated**, when it appears in C0 but disappears after the relevant context is introduced;
- **persistent**, when a gap observed in C0 reappears after relevant context is introduced, regardless of whether that context answers it partially or completely;
- **emergent**, when it does not appear in C0 but arises after context is added; or

- **unchanged**, when it appears in both conditions but the added context was not intended to resolve it.

The extracted questions were independently classified by three judges: one undergraduate student, one master's student, and one doctoral student. All three had prior experience in software engineering and requirements engineering research.

Inter-rater agreement was assessed using the original independent annotations, before majority voting or manual adjudication. Since three judges assigned categorical labels, agreement was measured using Fleiss' kappa. The resulting agreement was $\kappa = 0.238$ for contextual category, $\kappa = 0.069$ for informational coverage, and $\kappa = 0.318$ for functional relevance. The corresponding mean pairwise observed agreement rates were 55.9%, 66.4%, and 80.8%, respectively. Because coverage and relevance determine the N/U metric, the low pre-adjudication agreement represents an important construct-validity threat; the adjudicated labels should therefore be interpreted as a defensible coding rather than objective ground truth.

Final labels were subsequently assigned by unanimity when all judges agreed and by majority vote when two judges agreed. Cases without majority or involving coverage/relevance uncertainty were manually reviewed.

2.4 Metrics

The main outcome of interest is the number of informational gaps simultaneously classified as *necessary* and *unanswered*. This metric captures clarification needs that remain legitimate and unresolved in a given condition, and therefore serves as the primary indicator of whether a context condition reduces relevant informational deficiencies.

In addition, the study uses descriptive metrics derived from the protocol:

- **Elimination rate:** proportion of gaps observed in C0 that disappear after a contextual condition is introduced;
- **Category-specific elimination:** elimination behavior analyzed with respect to the intended informational function of each context type;
- **Redundancy rate:** proportion of gaps asking for information already explicitly available in the supplied material; and
- **Emergence rate:** proportion of gaps that arise only after additional context is provided.

3 Preliminary Results and Discussion

This section reports descriptive observations from the 24 executions after segmentation, normalization, classification, and adjudication of the extracted clarification questions. Given the exploratory design, the small number of User Stories, and the absence of repeated executions, the results should be interpreted as emerging evidence rather than conclusive findings.

3.1 Descriptive Overview

The 24 executions yielded 177 extracted question records. After removing 59 duplicates, 118 primary clarification questions remained (4.92 per execution; range: 3–5). Segmentation, semantic merging, and normalization yielded 115 condition-level gap occurrences, subsequently consolidated into 42 unique normalized informational gaps through cross-condition deduplication. Table 2 summarizes the results.

In Table 2, *Prim. Q.* denotes primary clarification questions, *Norm. Gaps* normalized informational gaps, *N/U* necessary and unanswered gaps, *Redund.* redundant gaps, and *Emerg.* emergent gaps.

Question counts alone are insufficient to evaluate clarification quality: fewer questions may coexist with unresolved necessary gaps, whereas additional questions may reflect dependencies or exceptions exposed by context. Interpretation therefore requires relating each normalized gap to its relevance and informational coverage.

Table 2: Preliminary descriptive results by context condition.

Cond.	Prim. Q.	Norm. Gaps	N/U	Redund.	Emerg.
C0	20	18	17	0	–
CL	20	20	15	0	8
CO	18	18	8	1	7
CD	20	20	16	0	9
CS	20	19	16	0	7
CT	20	20	7	2	10

Table 3: Category-specific elimination of gaps.

Gap category	CL	CO	CD	CS
Lexical	2	0	0	1
Operational	2	4	4	2
Decisional	2	2	2	2
Systemic	0	1	1	1
Unassociated	0	0	0	0

3.2 Clarification Needs Across Context Conditions

After normalization, the number of necessary and unanswered gaps changed from 17 in C0 to 15 in CL, 8 in CO, 16 in CD, 16 in CS, and 7 in CT. These values indicate mixed effects: operational and combined contexts produced the clearest reductions, while lexical, decisional, and systemic contexts produced only small reductions.

The combined context condition CT is particularly important because it represents the situation in which the agent receives all available contextual information. In this preliminary execution, CT produced 7 necessary and unanswered gaps. Compared with C0, this represents a reduction of 10 gaps (58.8%). CT produced the lowest observed N/U count (7); however, because CT combines all four context blocks, comparisons with isolated conditions also reflect a larger amount of supplied information. Among the isolated conditions, CO showed the lowest N/U count (8).

3.3 Category-Specific Effects

The second research question investigates whether each context category eliminates gaps associated with its expected informational function. The expected pattern is that lexical context should primarily reduce lexical gaps, operational context should reduce operational gaps, decisional context should reduce decisional gaps, and systemic context should reduce systemic gaps.

Table 3 presents the category-specific elimination results. The diagonal values represent eliminations by the corresponding context category. Off-diagonal values indicate cross-category effects, in which one context type eliminated a gap associated with another category.

Preliminary evidence indicates a partial correspondence between context categories and the gaps they eliminate. Operational gaps show this most clearly: operational (CO) and decisional (CD) contexts removed them equally often, four eliminations each, compared with two for lexical (CL) and two for systemic (CS) contexts. This overlap suggests that decisions about business rules often encode the same operational information that operational context describes directly, rather than a single category holding exclusive ownership over a given gap type. The off-diagonal eliminations, however, indicate that the four context categories are best treated as analytical dimensions rather than mutually exclusive classes. Such cases suggest that the categories are analytically useful, but not completely independent in practice.

3.4 Persistence, Redundancy, and Emergence

The third research question focuses on how gaps behave after contextual information is introduced. Three patterns are especially

relevant in this exploratory study: persistence, redundancy, and emergence.

A persistent gap occurs when a gap observed in C0 reappears after relevant context is introduced. Persistence is independent of coverage: the added context may partially answer the gap or make the corresponding question fully redundant. In the preliminary data, persistent gaps were observed in all contextual conditions: CL=2, CO=6, CD=1, CS=1, and CT=7. This indicates that context sometimes made relevant information available without preventing the agent from asking related clarification questions.

Persistence and redundancy sit on the same coverage scale defined in Section 2, but they capture different degrees of informational overlap. A persistent gap reappears in a compared condition while its coverage improves from unanswered to partially answered: the added context resolved part of the informational need without eliminating the gap entirely. A redundant question, by contrast, is one whose answer was already fully present in the supplied context. Of the 17 persistent-gap instances observed across conditions, 16 corresponded to a shift toward partially answered coverage, and only one corresponded to fully redundant coverage. This explains why persistence counts exceed redundancy counts in every condition: most persistence reflects incomplete rather than complete overlap with the context supplied.

Redundancy is analyzed within each condition by checking whether the answer was already available in the material supplied to the agent. It is relevant because a question can be functionally necessary while still being unnecessary to ask in that condition. For example, a question about an actor, rule, or limit may be important for the requirement, but redundant if the context already provides the answer. In the current data, the highest redundancy was observed in CT, with 2 redundant gaps, followed by CO with 1. For example, in US08, the agent asked whether a disconnected submitter's stored estimate should be revealed, even though the operational context stated that all stored estimates are revealed in a single broadcast event.

Emergent gaps occur when a question does not appear in C0 but arises after additional context is provided. These gaps should not be treated automatically as errors. In some cases, emergence may indicate that context enabled a deeper inspection of the requirement by exposing constraints, dependencies, or exceptions that were invisible in the User Story alone. In other cases, emergent questions may be unnecessary or outside the intended functional scope. In the preliminary data, emergent gaps were observed in every contextual condition: CL=8, CO=7, CD=9, CS=7, and CT=10.

3.5 Discussion and Implications

The results indicate that additional context does not simply translate into fewer questions: it may resolve gaps, expose new ones, or be insufficiently used by the agent. This supports normalized informational gaps, rather than question counts alone, as a more informative evaluation unit. For practitioners, a context-aware SDD agent should inspect available evidence and ask only for information that remains unresolved.

3.6 Threats and Limitations

The single-run design does not characterize LLM stochastic variability, while the five-question ceiling may cause question displacement and compress observed emergence or redundancy. Comparisons involving CT also conflate context diversity with context volume. Finally, the low pre-adjudication agreement for some dimensions indicates interpretive uncertainty, subsequently addressed through adjudication.

4 Final Remarks

This paper presented emerging results from a controlled exploratory study on context dependency in LLM-based requirements clarification. Four User Stories from different domains were analyzed under six context conditions, producing 118 primary questions across 24 executions.

The preliminary analysis suggests that clarification behavior should not be evaluated only by counting generated questions. Context may eliminate some clarification needs, but it may also lead to redundant, persistent, or emergent questions. The strongest preliminary reduction in necessary and unanswered gaps was observed under the combined-context condition (CT), followed by the operational context condition (CO), but CT also produced the highest number of emergent gaps.

The study is limited by the small number of User Stories, one execution per condition, one SDD agent, the five-question ceiling of the clarification workflow, and the reliance on human judgment during normalization and classification. Future work will expand the dataset, include repeated executions, and compare different models and agents.

Artifact Availability

The supplementary material, including the protocol, extracted data, and supporting materials, is publicly available in the replication repository.[1]

Acknowledgments

This work has been partially funded by the project "ISOP Engineering: Métodos e Ferramentas de Engenharia Inteligente para Sensoriamento Inteligente" 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.

Generative AI Use

Generative AI tools were used to support language editing. All scientific content was reviewed and approved by the authors, who take full responsibility for the manuscript.

References

[1] Anis Rahmawati Amna and Geert Poels. 2022. Ambiguity in User Stories: A Systematic Literature Review. *Information and Software Technology* 145 (2022), 106824. doi:10.1016/j.infsof.2022.106824

[2] Haowei Cheng, Jati H. Husen, Yijun Lu, Teeradaj Racharak, Nobukazu Yoshioka, Naoyasu Ubayashi, and Hironori Washizaki. 2026. Generative AI for Requirements Engineering: A Systematic Literature Review. *Software: Practice and Experience* 56, 2 (2026), 141–170. doi:10.1002/spe.70029

[3] Fabiano Dalpiaz. 2018. Requirements Data Sets (User Stories). Mendeley Data. doi:10.17632/7zbk8zsd8y.1

[4] Fabiano Dalpiaz, Ivor van der Schalk, Sjaak Brinkkemper, Fatma Başak Aydemir, and Garm Lucassen. 2019. Detecting Terminological Ambiguity in User Stories: Tool and Experimentation. *Information and Software Technology* 110 (2019), 3–16. doi:10.1016/j.infsof.2018.12.007

[5] GitHub. 2026. Spec Kit: A Specification-Driven Development Toolkit. https://github.com/github/spec-kit. Accessed: 2026-07-08.

[6] Kaicheng Huang, Fanyu Wang, Yutan Huang, and Chetan Arora. 2025. Prompt Engineering for Requirements Engineering: A Literature Review and Roadmap. In *2025 IEEE 33rd International Requirements Engineering Conference Workshops (REW)*. IEEE, 548–557. doi:10.1109/REW66121.2025.00081

[7] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. 2016. Improving Agile Requirements: The Quality User Story Framework and Tool. *Requirements Engineering* 21, 3 (2016), 383–403. doi:10.1007/s00766-016-0250-x

[8] Lloyd Montgomery, Davide Fucci, Abir Bouraffa, Lisa Scholz, and Walid Maalej. 2022. Empirical Research on Requirements Quality: A Systematic Mapping Study. *Requirements Engineering* 27, 2 (2022), 183–209. doi:10.1007/s00766-021-00367-z

[9] Mohammad Amin Zadenoori, Jacek Dąbrowski, Waad Alhoshan, Liping Zhao, and Alessio Ferrari. 2025. Large Language Models (LLMs) for Requirements Engineering (RE): A Systematic Literature Review. arXiv:2509.11446 [cs.SE] doi:10.48550/arXiv.2509.11446

[1] https://github.com/lucassa-bit/experimento-emerging-results