LLM-Based Reporting Assistant for Agile Project Management

Ioanna Plexida
Computer Engineering and Informatics Department,
University of Patras, 26504
Patras, Greece
ioannaplexida@ac.upatras.gr

Andreas Komninos
Computer Engineering and Informatics Department,
University of Patras, 26504
Patras, Greece
akomninos@ceid.upatras.gr

Abstract

We present an AI assistant that processes Jira data to generate concise sprint summaries, improving communication across Agile teams. The system is based on a hybrid two-layer architecture in which Layer 1 extracts verified metrics from Jira repositories to establish computable ground truth, and Layer 2 generates role-specific reports. We further introduce an evaluation framework based on text statistics and regular expressions (regex) to assess each report across seven independent dimensions, enabling comparison across different LLMs and sprint-size clusters. Experiments on real-world projects show that ticket count is the primary factor affecting report quality. Beyond summary generation, the proposed approach enables validation of report fidelity and can be integrated as a monitoring mechanism within Agile workflows, supporting both technical and business communication.

Keywords

Large language models, entity-level LLM evaluation, Agile project management, Jira, reference-free evaluation

1 Introduction

Large Language Models (LLMs) are increasingly used to automate Jira-based analysis [4, 6] and produce sprint reports [3]. However, their reliable evaluation remains challenging: reference-based methods require gold summaries that do not exist for sprint reports, while LLM-as-judge approaches introduce stochasticity, bias, and circular dependencies, along with significant computational costs that limit scalability [7].

We propose a hybrid two-layer architecture in which Layer 1 extracts verified metrics directly from Jira repositories to establish computable ground truth, and Layer 2 generates role-specific reports for key stakeholders, guided by role-specific prompts encoding personas, structural templates, grounding rules, and adaptive length budgets. Building on this, we introduce a multi-dimensional evaluation framework that assesses reports across coverage, numeric faithfulness, hallucination, structure, adaptive word count, critical coverage, and prompt-rule compliance. The framework extends claim-level factuality [5] and entity-based verification [2] to multi-role Agile reporting using structured project data.

Evaluation on real-world open-source datasets shows that ticket volume is the primary factor affecting report quality. Additionally, we observe both prompt-driven effects, where performance remains sensitive despite controlled prompt engineering, and model-driven trade-offs. The proposed approach is directly integrable into Jira pipelines, that can enable scalable threshold-based monitoring of LLM-generated reports.

2 Methodology

The proposed pipeline comprises five processing stages. Raw Jira ticket data from three open-source projects (Apache Kafka, SourceTree Windows, Apache ZooKeeper; 4,388 tickets spanning approximately 24 months per project) [1] are normalized and grouped into 14-day timebox windows. Layer 1 computes verified ground-truth metrics capturing key aspects of tickets activity, while Layer 2 generates role-specific reports for Scrum Master, Product Owner, Developer, and Client audiences using structured prompts with role, format, and length constraints. The full pipeline is illustrated in Figure 1. The separation of deterministic metric extraction (Layer 1) from generative narration (Layer 2) is motivated by the characteristics of our application domain. Jira tickets provide structured, numerically grounded data that can be leveraged to ensure the quality of generated reports, while multi-role reporting requires consistent communication across technical and business stakeholders, motivating the need for a unified and reliable generation framework. Layer 1 computes metrics directly from source JSON via fixed procedures, providing a stable and reproducible reference for evaluating Layer 2 outputs and avoiding the circularity of LLM-as-judge approaches. Numerical aggregation is handled by deterministic code, where LLMs are known to be unreliable, while linguistic synthesis and role-specific framing are delegated to Layer 2, where LLMs perform effectively. This separation introduces deterministic safety anchors that ensure reliability while enabling a model-agnostic design, making the framework compatible with different LLMs and robust to model updates, requiring only regeneration of Layer 2 outputs.

A unified single-pass design, in which an LLM consumes raw tickets and produces both metrics and narrative, is therefore avoided, as it conflates numerical errors with narrative hallucination and limits diagnostic traceability.

The evaluation module operates as an independent post-processor. Ground truth is defined as the set of metrics computed directly from the original Jira data, independent of any generated output. All models are evaluated under identical input conditions and prompt structures to ensure fair comparison. Each report is scored across seven dimensions. All metrics are computed at the report level and aggregated across sprint-size clusters using mean values to enable consistent cross-model comparison.The evaluation focuses on factual consistency and structural compliance rather than subjective semantic quality. **Coverage** and **Critical Coverage** measure ticket-entity recall via regex and set intersection (Blocker/Critical-filtered for the latter). **Numeric Faithfulness** verifies structured numeric claims via context-aware regex with bounded tolerance. **Hallucination** penalizes phantom IDs and out-of-set percentages. **Structure** checks header and per-role section presence. **Word Count** applies

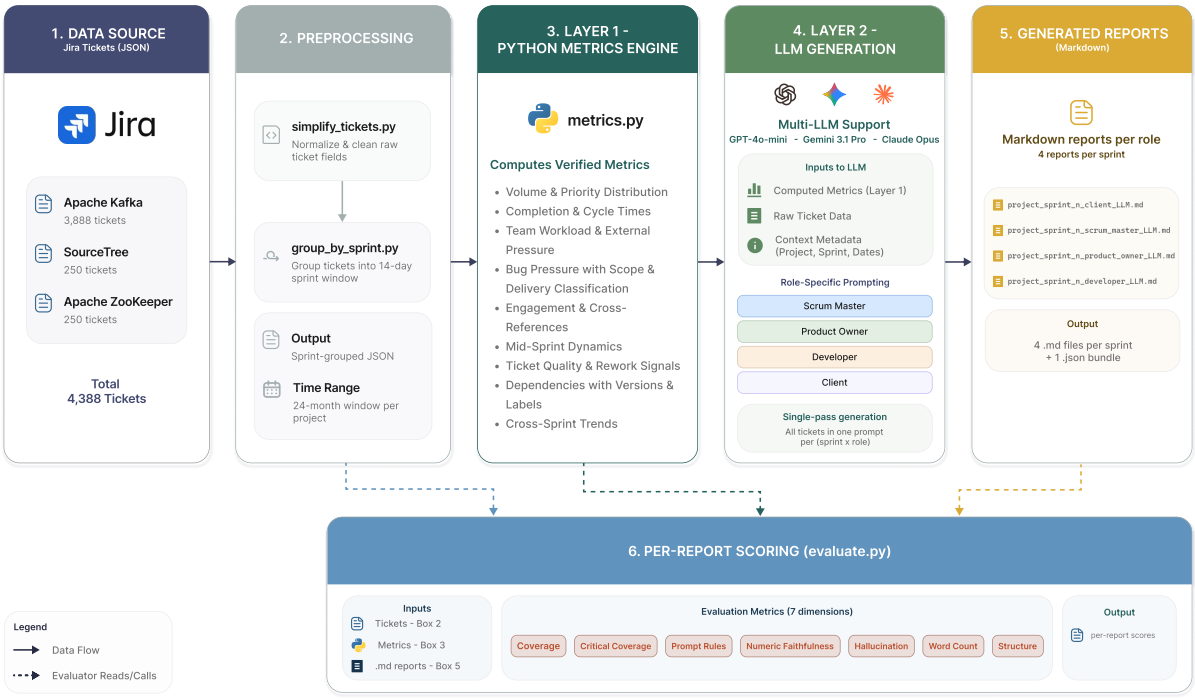

Figure 1: Hybrid 2-Layer Architecture for Reliable Reporting.

the adaptive limit with two-sided utilization scoring (base + per-ticket allowance). **Prompt Rules** combines banned-phrase, voice, readability (Flesch–Kincaid), and grounding-ratio checks.

Layer 2 prompts were engineered iteratively to balance role-appropriate content (e.g., business-value emphasis for Product Owner; technical detail for Developer; jargon-free narrative for Client) with length and structural constraints.

We focus the primary analysis on Apache Kafka (3,888 tickets across 53 sprints), with ZooKeeper and SourceTree used for cross-project generalizability. Three commercial LLMs (GPT-4o-mini, Claude Opus 4.6, Gemini 3.1 Pro) are evaluated on Kafka, yielding 320 generated reports across sprint-size clusters (Small, Medium, Large, and Extreme). K-means clustering (k=3) defines the primary sprint-size clusters based on ticket count: Small (14–66 tickets), Medium (68–112), and Large (133–181).

3 Results

We evaluate the performance of the generated reports based on the methodology described in Section 2 and illustrated in Figure 1. Figure 2 presents the mean per-cluster values across all seven evaluation metrics as a function of ticket count for three LLMs on the Kafka project. Additional experiments on ZooKeeper and SourceTree, which feature significantly lower ticket counts per sprint compared to Kafka, yielded strong performance across all metrics; however, detailed results are omitted due to page limitations.

Coverage and Critical Coverage curves for GPT-4o-mini and Gemini 3.1 Pro exhibit degradation as ticket count increases. In the large cluster both models stay within the word-count limit; however, in this regime GPT-4o-mini exhibits its weakest performance in structure, while Gemini 3.1 Pro underperforms in prompt-rule compliance. These observations confirm that ticket count is the dominant factor affecting report quality, while model-specific differences introduce additional variability.

Claude Opus 4.6 overshoots word-count targets in small and medium clusters, with its weakest performance observed in the medium cluster, leading to reduced Coverage and Critical Coverage. This reflects a systematic verbosity bias that is better accommodated under larger word-count budgets; consequently, performance improves at higher ticket counts, achieving higher Coverage and Critical Coverage, albeit with increased hallucination. Despite this, Claude demonstrates strong performance in prompt-rule compliance, structure, and numeric faithfulness. These model-driven differences suggest practical implications, indicating that model-specific length calibration may be required when deploying LLMs in environments with strict format constraints.

The persistent gap between Coverage and Critical Coverage, with the latter consistently higher, indicates that LLMs implicitly prioritize salient tickets under compression, mirroring human summarization behavior. Hence, the framework can act as a diagnostic tool, enabling practitioners to trace omitted tickets and assess whether such omissions are operationally acceptable.

4 Conclusion

We presented a hybrid two-layer approach for LLM-based Agile reporting that enables data-grounded evaluation. Our results show

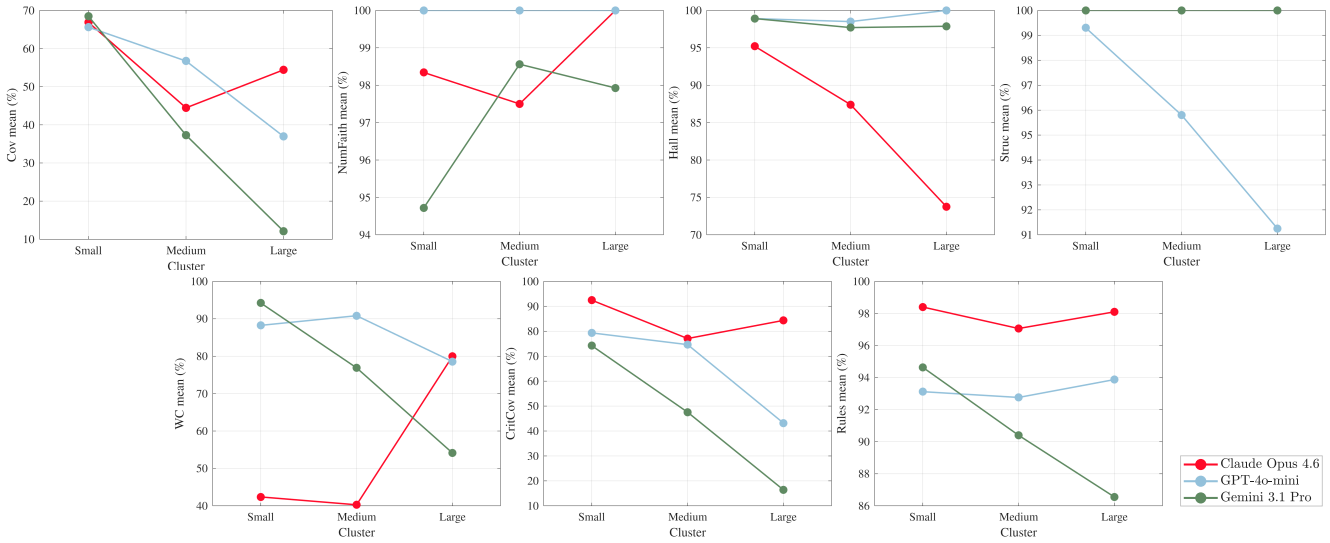

Figure 2: Per-cluster mean values of the seven evaluation metrics (Cov, NumFaith, Hall, Struc, WC, CritCov, Rules) across sprint-size clusters (Small, Medium, Large) for the three LLMs (GPT-4o-mini, Claude Opus 4.6, Gemini 3.1 Pro).

that ticket count is the dominant factor affecting report quality, while model-specific behaviors and prompt sensitivity introduce significant trade-offs. The consistent divergence between Coverage and Critical Coverage suggests that LLMs implicitly prioritize salient information under constrained generation. Future work will focus on systematic prompt-engineering iteration and incorporating complementary LLM-as-judge dimensions to capture semantic aspects.

References

[1] Lloyd Montgomery, Clara Lüders, and Walid Maalej. 2022. An alternative issue tracking dataset of public Jira repositories. In *Proceedings of the 19th International Conference on Mining Software Repositories*. 73–77.

[2] Feng Nan, Ramesh Nallapati, Zhiguo Wang, Cicero dos Santos, Henghui Zhu, Dejiao Zhang, Kathleen McKeown, and Bing Xiang. 2021. Entity-level factual consistency of abstractive text summarization. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume*. 2727–2733.

[3] Martin Schroder. 2023. Autoscrum: Automating project planning using large language models. *arXiv preprint arXiv:2306.03197* (2023).

[4] Amirali Shahriary, Mohammadsaeid Sedighi, Nima Tajik, Mohammadali Shahinfar, and Amirhossein Rahati Asiyabar. 2025. Assessing Large Language Models as Agile Scrum Masters: A Comparative Study of Project Planning Efficiency. In *2025 11th International Conference on Web Research (ICWR)*. IEEE, 150–156.

[5] Aivin V Solatorio. 2025. Proof-Carrying Numbers (PCN): A Protocol for Trustworthy Numeric Answers from LLMs via Claim Verification. *arXiv preprint arXiv:2509.06902* (2025).

[6] Yi Yao, Jun Wang, Yabai Hu, Lifeng Wang, Yi Zhou, Jack Chen, Xuming Gai, Zhenming Wang, and Wenjun Liu. 2024. BugBlitz-AI: an intelligent QA assistant. In *2024 IEEE 15th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 57–63.

[7] Yuwon Yoon, Kevin Iwan, Madeleine Zwart, Xiaohan Qin, Hina Lee, and Maria Spichkova. 2025. Scrum Sprint Planning: LLM-based and algorithmic solutions. *arXiv preprint arXiv:2512.18966* (2025).