Reporting Repository Analytics via Large Language Models: An Industrial Experience Report

Giovanni Rosa*, Juan José Arias Rojas* and Jesús M. González Barahona*

Universidad Rey Juan Carlos, Spain

Abstract

Mining Software Repositories (MSR) is a widely investigated area in software engineering, focusing on tools, processes, and methods aimed at extracting and analyzing data from the overall software development process. Recent advancements in software engineering related to the use of Large Language Models (LLMs), together with the increasing amount of data available for mining software repositories, open new opportunities to leverage the power of LLMs to automate the MSR process. This paper presents an industrial experience report exploring this direction. We present Compilanum, an LLM-based workflow for automatically generating qualitative reports from software repository activity metrics, providing insights related to the overall status, trends, project health, contributor activity, and sustainability of a software repository. We present an empirical evaluation of the factual accuracy of the generated reports and an expert evaluation of their actual usefulness, strengths and weaknesses, and potential for practical adoption in real-world scenarios. The results showed that Compilanum has the potential to generate accurate and useful reports, even if some inaccuracies are still present, making it a promising tool for supporting developers and decision-makers in understanding and managing software repositories.

Keywords

Software Analytics, Mining Software Repositories, Large Language Models, Empirical Software Engineering

1. Introduction

Large Language Models (LLMs) are changing the way we interact with software repositories and the overall processes of software development. In particular, LLMs have been applied in a wide range of software engineering tasks, including code generation, bug detection, and documentation generation [1]. Among other applications, software documentation and analytics are two areas where LLMs can provide significant benefits. In the past, extracting insights from software repositories has been a challenging task, requiring significant manual effort and expertise and a set of specialized tools to perform the retrieval, processing, and reporting of the data [2]. The discipline of Mining Software Repositories (MSR) has been extensively studied and developed to provide a set of practices and methodologies for extracting data from software repositories, but it is not exempt from challenges and issues related to data heterogeneity and bias [3]. MSR is being increasingly complemented by LLMs, which can help to automate part of the process and provide a more comprehensive understanding of software development processes and practices [4, 5, 6, 7]. In this context, we present an industrial experience report on the use of LLMs for automatically generating qualitative reports from software repository activity metrics, *i.e.,* commits, issues, pull requests, releases, and contributors. Specifically, starting from the data retrieved by *Perceval* [2], an open-source tool from the CHAOSS project [1], we use LLMs to generate a structured report that provides insights into the activity of a software repository, highlighting trends, patterns, and significant events that occurred during a specific time period. The goal is to spot trends and patterns in the repository activity that are not easily spotted with traditional data analysis techniques, and to provide a more comprehensive understanding of the repository's health and sustainability. We implemented Compilanum, a pipeline that feeds the data extracted from the repository into LLMs

Woodstock'22: Symposium on the irreproducible science, June 07–11, 2022, Woodstock, NY

*Corresponding authors.

giovanni.rosa@urjc.es (G. Rosa); juanjose.arias@urjc.es (J. J. A. Rojas); jesus.gonzalez.barahona@urjc.es (J. M. G. Barahona)

0000-0002-5241-1608 (G. Rosa); 0000-0001-9682-460X (J. M. G. Barahona)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1] https://www.chaoss.community/

in order to summarize several aspects of the repository activity, covering the overall status, trends, project health, contributor activity, and sustainability. To evaluate the effectiveness and the practical usefulness of Compilanum, we conducted an empirical evaluation with three open-source software repositories, selected from the GitHub platform, and a semi-structured interview with experts from Bitergia, a company that provides software analytics services and is actively involved in the development of the CHAOSS project. Specifically, we evaluated the factual accuracy, relevance, and insightfulness of the information provided in the generated reports, as well as its congruence with the underlying data, leveraging the study participants' direct knowledge of the analyzed projects. The results of the evaluation show that Compilanum is able to generate reports with a high level of factual accuracy, even if some inconsistencies are still present. However, the experts provided positive feedback on the content of the reports, highlighting the ability of the LLM to spot very in-depth insights that are not evident with the normal techniques for repository analytics. However, the evaluation is limited to the specific study context, and the results may not be generalizable to other application domains or large projects. Further research is needed to extend the evaluation and assess the overall effectiveness of Compilanum in different contexts, as well as its scalability and performance when applied to larger repositories or different types of software projects. This means that Compilanum can be a valuable tool in real-world scenarios, helping both developers and decision-makers gain a better understanding of a software project's activities and make informed decisions. However, it still needs to be improved in order to provide more contextual information and to reduce the presence of inaccuracies in the generated reports.

The rest of the paper is structured as follows. Section 2 presents the related work. Section 3 describes the workflow of Compilanum. Section 4 and Section 5 present the design and the results of the empirical evaluation. Section 6 provides a set of lessons and implications of the findings, and Section 7 presents the threats to validity. Finally, Section 8 concludes the paper.

2. Related Work

Mining Software Repositories (MSR) is a set of practices and methodologies aimed at extracting data during software development, such as source code, commits, issues, and pull requests, to derive actionable insights about software products and processes. GitHub is one of the most widely used sources of information, as it exposes a large number of repositories, particularly in the open-source domain, providing a rich source of data for MSR studies. However, the use of GitHub data also presents challenges: repository-selection choices, incomplete activity records, and misleading platform metadata can bias the resulting analyses [3]. Toolsets such as GrimoireLab operationalize MSR by supporting the retrieval, analysis, visualization, and reporting of data from heterogeneous software repositories, as well as several data-enrichment techniques [2].

Recent work increasingly uses LLMs to interpret this heterogeneous data. A survey of 177 studies identifies classification, generation, detection, and information extraction as the most common applications of language models in MSR, with GitHub being the dominant data source [4]. Another example is the study by Simoes *et al.* [8], which used LLMs to assess source-code quality. The authors found that LLMs capture characteristics complementary to those measured by traditional static-analysis tools. De Martino *et al.* [6] used an LLM to detect documented and previously unknown green architectural tactics in GitHub projects. They also proposed the PRIMES framework [5], which helps formalize the process of LLM-assisted MSR by providing a structured approach to prompt refinement, model comparison, and output validation for LLM-based repository-mining studies. More generally, LLMs can transform unstructured text into structured knowledge with limited supervision [9]. These studies show that LLMs are effective at extracting and interpreting information from software repositories. Together with traditional MSR techniques, they can provide a more comprehensive understanding of software development processes and practices.

Several studies have also explored the use of LLMs in automated software documentation. A recent literature review reports applications including code summarization, comment and commit-message

generation, requirements engineering, and technical-document analysis [10]. An example is RepoAgent [11], which works at the repository level, generating and maintaining documentation based on the code structure and reference relationships.

Whereas those works mainly focus on code-level documentation, our work aims to join the MSR and LLMs research lines by using LLMs to generate qualitative reports from software repository activity metrics, providing a high-level overview of the repository's activity and insights into its development process.

3. Automated Report Generation with Compilanum

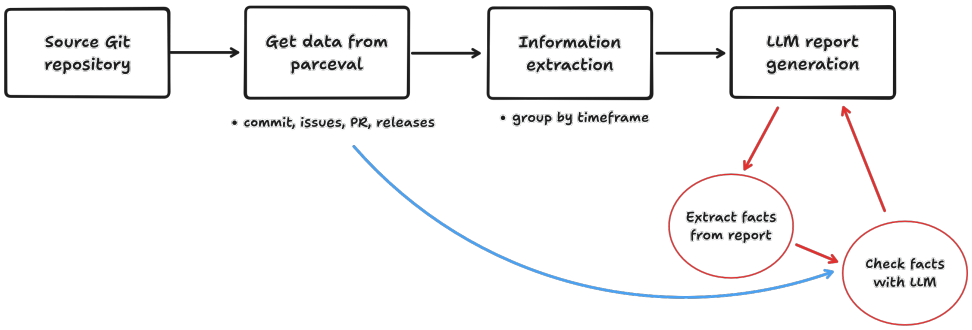


Figure 1: Workflow of Compilanum.

In this section, we present Compilanum, our approach for automatically generating qualitative reports from software repository activity metrics. The main goal is to provide users with a comprehensive and insightful summary of the activity of a software repository, based on the data extracted from the repository itself, without requiring users to manually analyze the data or to have expertise in interpreting software repository metrics. Fig. 1 shows the steps that compose the workflow of Compilanum. It consists of three steps: (i) data extraction, (ii) report generation, and (iii) fact extraction and verification.

In the first step, we mine software repositories using *Perceval*[2], an open-source tool from the CHAOSS project[3] that provides a unified interface for retrieving data from software-development platforms such as GitHub, GitLab, and Bitbucket. Compilanum relies on *Perceval* to mine data describing the overall activity of GitHub repositories in terms of commits, issues, contributors, releases, and pull requests.

These data are then used as input for the subsequent report-generation step. For the one-year period considered in our study, Compilanum initially partitions the extracted data into three-month windows and prompts a Large Language Model (LLM) to generate a partial report for each window. If the data in a window exceed the amount that can be processed by the LLM, Compilanum automatically divides that window into progressively smaller intervals until each input is manageable. Once the partial reports cover the entire target period, the LLM synthesizes them into the first version of the report. The workflow uses zero-shot prompting both to generate partial reports from the extracted repository data and to synthesize those partial reports into a structured report. The result is a Markdown document organized into six main sections: (i) an executive summary presenting the overall repository status and key findings; (ii) a trends section describing how the repository's activity and development priorities evolved during the analyzed period; (iii) a section assessing project health through aspects such as issue handling, release stability, development pace, and technical concerns; (iv) a section describing contributor activity, roles, turnover, and the concentration of code ownership; (v) a section for sustainability assessment, discussing community continuity, long-term risks, and opportunities; and (vi) a conclusion section summarizing the repository's condition and providing actionable recommendations. The generated report is intended to provide qualitative insights into the activity of the repository, highlighting trends,

[2]https://github.com/chaoss/grimoirelab-perceval
[3]https://chaoss.community/

patterns, and significant events that occurred during the target period and that are not easily spotted by looking at the data.

Finally, in the last step, COMPILANUM performs a fact extraction and verification process to improve the accuracy and reliability of the generated report. This step involves (i) extracting a list of factual statements contained in the report, and (ii) verifying them against the underlying data extracted from the repository. For each extracted statement, the pipeline assesses whether it is supported by the repository data. Statements found to be inaccurate or unsupported are flagged as potential hallucinations, and the LLM is prompted to correct or remove them from the report. This process is automatically performed by another LLM, which analyzes the generated report and the underlying data to identify discrepancies between them, implementing the LLM-as-Judge pattern [12].

Because this automatic revision is a best-effort process, partially supported or incorrect statements may remain in the final report. Nevertheless, the process helps reduce potential hallucinations and improve the accuracy and reliability of the final output.

To assess the overall reliability of the approach and the accuracy of the generated reports, we performed several rounds of generation, feedback, and correction using data from the *kubeflow* repository, one of the repositories included in our study context. This allowed us to iteratively refine the prompts and the report-generation process with support from one of the Bitergia developers. In each round, we (i) generated the report and (ii) requested a manual review to obtain feedback on the accuracy and relevance of the information presented. This process was repeated several times until we obtained a satisfactory report containing accurate and relevant information. The use of the reports generated for *kubeflow* to tune and validate the prompts and the overall report-generation pipeline served as a pilot study.

In this procedure, we also fixed the underlying LLM prompts and settings. Specifically, we used one of the most powerful open-weight LLMs available at the time of conducting the study, *i.e., openai/gpt-oss-120b* [4], a 120B-parameter model released as the open alternative to the closed-source GPT models from OpenAI. We used *OpenRouter* [5] as the inference provider, leaving all parameters at their default values except for the temperature, which we set to 0.5 for the report generation and combination phases and to 0.0 for the fact extraction and verification phase.

4. Empirical Study Design

The *goal* of this study is to evaluate the effectiveness of COMPILANUM in automatically generating qualitative reports from software repository activity metrics, specifically focusing on the accuracy and relevance of the extracted factual information. Our study is guided by the following research questions:

- RQ_1: *To what extent are the facts generated in the report correct?* The aim of this RQ is to assess the factual accuracy of COMPILANUM by verifying whether the generated statements correctly reflect the underlying data of the software repository.
- RQ_2: *To what extent are the reported facts relevant and insightful?* This RQ evaluates the utility of the information provided. We aim to validate, through expert judgment, if the reports provide non-trivial qualitative insights that are useful for understanding repository activity.

4.1. Context Selection

The *context* of our study is composed of 3 open-source software repositories selected from the GitHub platform. Specifically, we selected repositories with which Bitergia developers are familiar in order to assess the relevance of the generated reports with the help of experts. In detail, we selected two projects from *GrimoireLab*, *i.e., grimoirelab-core*[6] and *grimoirelab-perceval*[7], and *kubeflow*[8]. Table 1 summarizes

[4] https://huggingface.co/openai/gpt-oss-120b
[5] https://openrouter.ai/openai/gpt-oss-120b
[6] https://github.com/chaoss/grimoirelab-core
[7] https://github.com/chaoss/grimoirelab-perceval
[8] https://github.com/kubeflow/kubeflow

Table 1
Repository demographics extracted from January to December 2025. The values are computed from raw Perceval data.

	Commits				Issues				Pull Requests			
Repository	**Total**	**# Contrib.**	**/month**	**Tags**	**Total**	**Open**	**Closed**	**Med. close (d)**	**Total**	**Open**	**Merged**	**Med. merge (h)**
grimoirelab-core	234	4	19.5	4	13	2	11	7.97	94	3	88	23.1
grimoirelab-perceval	91	4	7.6	31	6	3	3	0.55	29	8	14	16.9
kubeflow	36	14	3.3	4	19	0	19	1.02	120	0	23	38.3

the repository demographics using the data mined by *Perceval*. The time interval is January–December 2025. The repositories present different activity profiles: *grimoirelab-core* recorded the most commits (234), *grimoirelab-perceval* showed the highest release cadence (31 tags), and *kubeflow* received the most pull requests (120) and involved the largest number of contributors (14). This allows us to evaluate Compilanum across projects with different levels and forms of development activity.

4.2. Experimental Procedure

To answer RQ1, we first generate the report for each repository included in the study context. The main risk in this process is that the generated report might contain hallucinations, *i.e.,* statements that are not supported by the underlying data. To mitigate this risk, we extract the facts from the generated report and verify them against the underlying data from the repository. This process allows us to measure the reliability of Compilanum in generating accurate reports and to identify any potential hallucinations that might be present in the generated reports. The generation process involves the full pipeline of Compilanum, which means that (i) we extract all the metrics from GitHub using *Perceval* (*i.e.,* commits, issues, contributors, and pull requests), (ii) Compilanum generates a first version of the report from the extracted metrics, and (iii) we perform fact extraction and verification, leading to the corrected and final version of the report. We consider a 12-month target period for each generated report. As described in Section 3, Compilanum initially divides the repository data into three-month windows and automatically uses smaller intervals when the data in a window exceed the amount that can be processed by the LLM. Furthermore, for each generated report, we manually verify the correctness of the generated facts by comparing them with the underlying data from the repository. In detail, with the help of LLMs, we extract the list of factual statements contained in the report and, for each factual statement, we manually review it and verify if it is supported by the data extracted from the repository. We compute, as a result, the percentage of correctly generated facts for each report, providing an overall measure of the factual accuracy of Compilanum in generating reports.

To answer RQ2, we conduct semi-structured interviews with experts from Bitergia to assess the relevance, insightfulness, clarity, and practical usefulness of the reports generated by Compilanum. Our research protocol is informed by established guidelines for case study research in software engineering [13]. The participants have substantial experience in open-source repository analysis and are familiar with the selected projects, enabling them to provide informed feedback on the generated reports.

For this evaluation, we consider the reports generated for *grimoirelab-core* and *grimoirelab-perceval*. We exclude *kubeflow* because it was used during the development of Compilanum to tune and validate its prompts. Each report summarizes repository activity over a 12-month period. We provide participants with two versions of each report, generated by adding an instruction that specifies the target audience and influences the report's tone and level of detail. The *technical* version provides a detailed account for developers and prospective contributors. The *executive* version provides a more concise account for project managers, board members, and other decision-makers interested in project sustainability and strategic dependencies. This evaluation examines whether the reports are useful and insightful for these different target audiences and elicits potential improvements to the report-generation process.

The interviews are conducted in English via video conference, and each interview lasts approximately 60 minutes. With the participants' consent, the interviews are recorded and transcribed for subsequent

Table 2
The semi-structured interview structure with questions, ratings, and optional probes.

ID	Type	Question, Scale, and Probes
Phase 1: Introduction and participant context		
Q1.1	Open	**Could you briefly describe your relationship with the project and how familiar you are with its recent activity?**
Q1.2	Likert	**How familiar are you with the project analyzed in the report?**
	Scale	*1: Not familiar at all; 2: Slightly familiar; 3: Moderately familiar; 4: Very familiar; 5: Expert, core contributor, or long-term maintainer.*
	Probe 1	*What type of activities have you performed in the project?*
	Probe 2	*How closely do you follow its development?*
	Probe 3	*Which parts of the project do you know best?*
Phase 2: Overall impression and usefulness of the report		
Q2.1	Open	**What was your overall impression of the report, and how useful would it be in your work?**
	Probe 1	*Which information or section was the most useful?*
	Probe 2	*Which information or section was the least useful?*
	Probe 3	*Did the report help you notice something that was not immediately visible from the raw data or from the information provided by GitHub?*
	Probe 4	*Who do you think would benefit most from this report?*
	Probe 5	*Can you give a concrete example of a task, decision, or situation in which you could use the report?*
Q2.2	Likert	**How would you rate the overall usefulness of the report?**
	Scale	*1: Not useful; 2: Slightly useful; 3: Moderately useful; 4: Very useful; 5: Extremely useful.*
Phase 3: Insights, clarity, and presentation		
Q3.1	Open	**How easy was the report to understand, and did it provide any meaningful insights that you did not already have?**
Q3.2	Likert	**How easy was it to understand the metrics, charts, explanations, and conclusions?**
	Scale	*1: Very difficult; 2: Difficult; 3: Neither difficult nor easy; 4: Easy; 5: Very easy.*
Q3.3	Likert	**To what extent did the report provide useful information that you did not already know or could not easily infer from the repository data?**
	Scale	*1: No new insights; 2: Few new insights; 3: Some new insights; 4: Several useful new insights; 5: Many useful new insights.*
	Probe 1	*What was the most useful or surprising insight?*
	Probe 2	*Were any explanations unclear or too technical?*
	Probe 3	*Was the level of detail appropriate?*
Phase 4: Accuracy and quality of the conclusions		
Q4.1	Open	**Based on your knowledge of the project and the evidence presented, how accurate and well supported did the report appear?**
	Probe 1	*Did you notice any factual errors?*
	Probe 2	*Did the report draw conclusions that were too strong for the available evidence?*
Q4.2	Likert	**How accurate did the factual information appear?**
	Scale	*1: Mostly inaccurate; 2: Several important inaccuracies; 3: Generally accurate, with some issues; 4: Accurate, with minor issues; 5: Completely accurate; Unable to assess.*
Phase 5: Evaluation of the report-generation approach		
Q5.1	Likert	**Overall, how valuable do you consider this approach?**
	Scale	*1: Not valuable; 2: Slightly valuable; 3: Moderately valuable; 4: Very valuable; 5: Extremely valuable.*
Q5.2	Likert	**Assuming that the verification process works as described, how much would it increase your trust in the report?**
	Scale	*1: It would not increase my trust; 2: It would increase my trust slightly; 3: It would moderately increase my trust; 4: It would substantially increase my trust; 5: It would make the report highly trustworthy.*
	Probe 1	*Is the verification process reflected in the final report?*
	Probe 2	*If the verification process were not present, would you trust the report?*
Phase 6: Final improvements and adoption		
Q6.1	Open	**What is the single most important improvement that would make the report or the overall approach more useful?**
	Probe 1	*What should be added to the report, the approach, or both?*
	Probe 2	*What should be removed?*
Q6.2	Open	**Assuming that the approach runs automatically, over which time windows should the reports be generated (e.g., the last six months or the last year)?**
	Probe 1	*Why?*
	Probe 2	*How could the appropriate time window be defined?*
Closing questions		
Q7.1	Open	**Is there anything important about the report or the proposed approach that we have not discussed?**
Q7.2	Open	**A state-of-the-art way of implementing this approach is to use multiple agents. Do you think multiple agents should be included, or is the current approach sufficient?**

analysis. One author conducts the interviews using a predefined guide containing the questions and optional probes; all authors contribute to the design and review of this guide.

The interview comprises six main phases followed by a closing discussion. In the first phase, we collect contextual information about each participant's experience with and knowledge of the project. In the second and third phases, participants provide their overall impression of the reports and assess their usefulness, along with the clarity of the provided information and whether the reports provide insights that cannot be easily obtained from the repository data or GitHub. In the fourth phase, participants assess the reports' perceived factual accuracy and whether their conclusions are adequately supported by the available evidence. This subjective assessment complements, rather than replaces, the manual data-based verification performed for RQ1. The fifth phase evaluates the perceived value of the report-generation approach and the extent to which its factual-verification procedure affects participants' trust in the reports. In the sixth phase, we collect suggestions for improving the reports and discuss potential adoption scenarios, such as using different time windows or generating reports continuously to support project monitoring. Finally, participants can raise issues not covered by the preceding questions.

The interview combines open-ended questions with five-point Likert-scale questions. The Likert-scale questions capture participants' familiarity with the project and their perceptions of the reports' usefulness, clarity, insightfulness, factual accuracy, value, and trustworthiness. The open-ended questions and optional probes guide the discussion and allow participants to provide additional context, examples, concerns, and suggestions for improvement. Table 2 reports the complete list of questions, and the interview guide is included in the replication package [14].

We analyze and report the interview results using a mixed-methods approach, combining a quantitative analysis of the Likert-scale responses with a qualitative analysis of the open-ended responses.

4.3. Data Availability

We provide all the data and scripts required to replicate our results in our replication package [14].

5. Results

5.1. RQ1: Factual Accuracy of the Generated Reports

Table 3
A summary of the factual accuracy of the generated reports.

	Technical				Executive			
Repository	**# Facts**	**Correct**	**Partial/likely**	**Incorrect**	**# Facts**	**Correct**	**Partial/likely**	**Incorrect**
grimoirelab-core	32	23 (0.72)	**7 (0.22)**	**2 (0.06)**	**40**	27 (0.68)	**12 (0.30)**	**1 (0.03)**
grimoirelab-perceval	**44**	**35 (0.80)**	8 (0.18)	1 (0.02)	32	28 (0.88)	4 (0.13)	0 (0.00)
kubeflow	23	18 (0.78)	**5 (0.22)**	0 (0.00)	35	**32 (0.91)**	3 (0.09)	0 (0.00)

The first RQ aims to assess the overall reliability of Compilanum by quantitatively measuring the factual accuracy of the generated reports. Table 3 describes the facts extracted from the generated reports and their validation outcomes using the LLM-assisted manual verification process. The table reports the total number of facts extracted, along with the number of correct facts, partially correct facts, and incorrect facts for each report. Each repository is evaluated for the two report profiles, technical and executive. Specifically, *parceval* has the reports containing the most correct facts overall (44 and 32), while *kubeflow* is the report with the least facts (23 and 35). The first is also the one having the highest number of correct percentage of facts for both of the report profiles (80% for technical and 88% for executive). On the other hand, *grimoirelab-core* has a larger number of incorrect facts reported. Note that this is the final evaluation after the fact-checking step, which means that the incorrect facts have been already mitigated in the fact-checking phase. Overall, the factual accuracy of the generated reports is high, although not perfect. This can be explained by the fact that the reports are generated

from varying amounts of commit data. For example, *grimoirelab-core* is the biggest repository in terms of commits, but in terms of the total number of facts and correctness, *perceval* is the best one. There may be other intrinsic factors influencing the factual accuracy of the generated reports, which we will investigate in future work. Overall, the results of RQ1 show that COMPILANUM is able to generate reports with a high level of factual accuracy, which is essential for ensuring the reliability and trustworthiness of the generated reports.

5.2. RQ2: Semi-structured Interview with Experts

Table 4
Likert-scale ratings from the two interview sessions. Shared ratings are those provided by both participants regarding both report profiles (*i.e.,* technical and executive). P indicates the specific score provided for *grimoirelab-perceval* and C for *grimoirelab-core*.

		Session 1			Session 2		
Question	**Construct**	**Technical**	**Executive**	**Shared**	**Technical**	**Executive**	**Shared**
Q1.2	Project familiarity	–	–	P: 5; C: 4	–	–	4
Q2.2	Usefulness	3	5	–	5	3	–
Q3.2	Clarity	3	5	–	–	–	5
Q3.3	New insights	–	–	5	4	3	–
Q4.2	Factual accuracy	–	–	4	–	–	4
Q5.1	Approach value	–	–	4	–	–	3
Q5.2	Trust from verification	–	–	4	–	–	5

For the semi-structured interviews conducted in the context of RQ2, we held two sessions with two participants, both from Bitergia, assessing the reports generated for *grimoirelab-core* and *grimoirelab-perceval* in the two different report profiles (*i.e.,* technical and executive). The two participants provided complementary perspectives on the analyzed projects: Participant 1 had approximately six years of experience as a data analyst and consultant, preceded by three years as a software developer. He was a former Perceval contributor and remained familiar with both projects through their use and participation in development meetings. Participant 2 had approximately four years of engineering leadership experience, leading teams responsible for developing and deploying the tools. Although he had not directly contributed to the code development, he was part of the development process in feature-planning discussions and has experience using both projects in several activities. In terms of quantitative measures, their self-reported familiarity ranged from 4/5 to 5/5. Table 4 reports the Likert-scale ratings collected during the interviews. The table shows the ratings provided by each participant for both the technical and executive report profiles of the two analyzed projects. There are some differences in the ratings provided by the two participants, which can be attributed to their different roles and perspectives. For example, Participant 1 rated the technical report as less useful and clear than the executive report, while Participant 2 rated the technical report as more useful and insightful than the executive report. These differences highlight the importance of considering the target audience when generating reports and tailoring their content and presentation accordingly. In general, the reports were perceived as useful and insightful, with ratings ranging from 3/5 to 5/5 across the different constructs. Regarding usefulness (Q2), both participants provided additional feedback, not readily visible on GitHub, regarding contributor roles, role transitions, development trends, and project risks. On the contrary, they disagreed on the preferred format: Participant 1 favored the executive report, whereas Participant 2 favored the technical report. Regarding clarity and insightfulness (Q3), the reports were generally understandable, but numerical information in the technical report lacked visual comparisons, while the executive report was considered too technical for decision-makers. Regarding accuracy (Q4), both participants assigned an overall rating between 4 and 5: the role of the contributors and historical information closely matched their knowledge, but they reported the presence of some inconsistent claims (*i.e.,* not directly backed by the facts), and some language artifacts were identified (*e.g.,* mixing English with other languages, some auto-corrected sentences). Regarding the approach and verification

process (Q5), participants considered automated fact checking essential for trust, while emphasizing that data quality, identity resolution, and human review remained necessary. Regarding improvements and adoption (Q6), they recommended adding charts, a structural overview, audience-specific indicators, richer review data, and adaptable reporting windows. Finally, regarding the architecture (Q7), one participant supported investigating specialized agents for greater control, while the other preferred refining the existing pipeline, which he considered already close to practical use. This is particularly important because it means that even if there are still some points to improve, the generated reports are already useful and insightful for the experts, and they are potentially production-ready artifacts usable in practice with real customers and stakeholders.

6. Discussion

The results of the experience report highlight the potential of Compilanum to generate automated reports that provide useful insights with a notable level of factual accuracy.

The implications of these findings are significant for both researchers and practitioners in the field of software repository analytics. For researchers, although this study is subject to the limitations of an experience report, it shows that LLMs have the potential to automate the generation of reports, providing a valuable tool for analyzing and understanding software repositories. For practitioners, the findings suggest that Compilanum can be a valuable tool in practice. Specifically, given the two profiles of the generated reports, technical and executive, the reports can support developers who want to start contributing to a new project or guide them toward maintenance activities and help them spot trends. For managers, stakeholders, and decision-makers, the executive reports can support decision-making processes related to project sustainability, resource allocation, and strategic planning. They can also help identify potential risks and opportunities when deciding to use such a project in their own software ecosystem, including weaknesses in long-term maintenance and continuity and possible supply-chain risks.

Regarding the limitations, it is important to note that the evaluation was conducted with a limited number of repositories and experts, which may limit the generalizability of the findings. Nevertheless, the aim of the presented evaluation is not to provide a generalizable assessment of the effectiveness of Compilanum, but rather to provide an industrial experience report of a specific application context. A potential limitation is also related to the scale of the analyzed software projects. For huge repositories with a large amount of development data, the current approach may not be able to generate a comprehensive report because it was designed as a proof of concept to show the potential of LLMs in this context rather than for production scalability. Further research is needed to assess the scalability and performance of Compilanum when applied to larger repositories or different types of software projects.

However, the evaluation also reveals areas for improvement and some challenges that need to be addressed in order to make Compilanum a robust tool for supporting the day-to-day work. In summary, we can draw the following main lessons learned from the conducted evaluation:

Missing contextual information: The main issue that arose from the evaluation is the lack of contextual and domain information for the generated reports. Since the reports are generated on the basis of repository analytics, the factual information reported is confined to the local environment of the repository, without considering the broader context in which the repository is used. For example, the repository may be part of a larger ecosystem or may be at a particular stage of its development (*e.g.,* a young repository with new features or a mature project with a stable codebase and routine maintenance activities). Since this information is not captured, there can be factual inaccuracies related to the sustainability of the project or the effective maintenance state of the analyzed project. To complete the current pipeline, contextual information regarding the software ecosystem in which the specific repository is contained is necessary to provide more in-depth insights in the final report, capturing additional information that can be difficult to infer from repository analytics alone.

Limits of the models used: The evaluation also highlighted some limitations of the LLM used in the Compilanum pipeline. One of the main issues is the hallucination of facts, which can lead to

the generation of incorrect information in the final report. This is mitigated by the fact-checking step, but it is still imperfect since there are still some partially correct facts that are not fully supported by the underlying data (see Table 3). More evident are the practical limits of LLMs in terms of context size, which can lead to the need to split the input data into smaller chunks when covering larger time windows. This must be done carefully to avoid losing the context of repository activity, which can lead to the generation of incomplete or misleading information in the final report. A possible solution would be to use a supporting database with advanced textual filtering and retrieval capabilities, such as OpenSearch, to better navigate the input metrics, together with a multi-agent architecture in which different agents can process different parts of the input data and generate a more comprehensive report.

🔍 **Additional pre-processing needs**: Some information extracted from commit activity, on the other hand, still needs preprocessing. The most evident example concerns contributions made to the repositories. This preprocessing applies to alias resolution, which identifies the same contributor under different names or email addresses, and to contributor roles, which are not always clear from commit activity alone. It is also important to distinguish among contributions made by bots, humans, and coding agents. GitHub provides some form of identification for some of them, but a future improvement would be to add a preprocessing step to identify contributors in terms of aliasing and contributor type in order to avoid inconsistencies and inaccuracies in the final report.

7. Threats to Validity

Construct Validity. We evaluated the generated reports in terms of the factual accuracy of the facts reported in the final report. However, this may not be completely representative of the overall project activity, and some of the information may be missing or incorrect, or the LLM may produce hallucinated conclusions. We mitigated this by verifying the facts with the underlying data, but this still represents a threat in terms of the completeness of the represented information. In addition, we integrated the semi-structured interviews with domain experts to obtain a more complete assessment of the generated reports, in order to have a more complete evaluation of the overall quality of the generated reports.

Internal Validity. The fact-validation step of the pipeline could have been affected by biases related to human subjectivity and LLM hallucinations. We mitigated this threat by performing a double validation of the facts using both LLM-assisted and manual verification. This reflects the fact that the majority of the reported facts are correct and backed by the underlying data. For the semi-structured interviews, we excluded the *kubeflow* repository from the evaluation since it was used to tune the prompts and validate the pipeline during the development of Compilanum. This mitigated the risk of biasing the evaluation results, but it also limits the generalizability of our findings to other repositories. The RQ1 result should still be viewed as post-tuning evidence rather than an independent evaluation. Another threat is represented by the influence and variability of the LLM used in the pipeline. We mitigated this threat by using a state-of-the-art open-weight LLM, but we acknowledge that different LLMs can produce different results and the final quality can vary. However, using one of the available frontier LLMs should provide a good baseline for the evaluation, and we expect that the results can be replicated with other LLMs with similar capabilities.

External Validity. Our evaluation was conducted in a specific industrial and open-source context. It included three GitHub repositories, two from the same GrimoireLab ecosystem, and examined a single one-year period. The limited number of repositories may limit the generalizability of the proposed findings. However, the selected repositories were specifically chosen to be familiar to the experts involved in the evaluation in order to provide a more complete and informed assessment of the generated reports. A larger number of repositories and expert evaluators would be needed to provide a more comprehensive evaluation of the proposed approach and to assess its generalizability across different contexts and domains. This is left as future work. We provide all the data and scripts required to replicate our results in our replication package [14]. The current study is an experience report based on a controlled, domain-specific evaluation of the proposed approach, which can be used as a baseline for future evaluations in different contexts and domains.

8. Conclusion and Future Work

This paper presents an industrial experience report on the automated generation of reports for software repository analytics using Large Language Models (LLMs). Specifically, we defined Compilanum, a pipeline that leverages LLMs to generate reports from activity data mined from Git repositories, including a validation step to ensure the factual accuracy of the generated reports. We evaluated Compilanum in an industrial context, using three open-source GitHub repositories and two domain experts from Bitergia, a company specialized in software analytics services. The evaluation focused on assessing the factual accuracy of the generated reports and gathering qualitative feedback from the experts regarding their usefulness and relevance. The results show that Compilanum can generate reports with a good level of factual accuracy. The experts provided positive feedback regarding the reports' usefulness and relevance but also identified limitations and inaccuracies to be addressed in future work. Overall, this experience report provides valuable insights into the potential of LLMs for automating the generation of reports for software repository analytics, helping to spot trends and patterns that are not easily identifiable by looking at the raw data. Future work will focus on addressing the identified limitations, improving the prompts and the LLMs used in the pipeline, and performing a more extensive evaluation to assess the overall generalizability of the approach across different contexts and domains.

Data Availability

The replication package with the scripts, interview guide, and data used in this study is publicly available [14].

Acknowledgment

This work has been partially supported by the Spanish Ministry of Science and Innovation under Grant No. 2024/00416/002. We thank the Bitergia team for their valuable support and collaboration in the study.

References

[1] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, H. Wang, Large language models for software engineering: A systematic literature review, ACM Transactions on Software Engineering and Methodology 33 (2024) 1–79.

[2] S. Dueñas, V. Cosentino, J. M. Gonzalez-Barahona, A. d. C. San Felix, D. Izquierdo-Cortazar, L. Cañas-Díaz, A. P. García-Plaza, Grimoirelab: A toolset for software development analytics, PeerJ Computer Science 7 (2021) e601.

[3] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, D. Damian, The promises and perils of mining github, in: Proceedings of the 11th working conference on mining software repositories, 2014, pp. 92–101.

[4] M. Romero-Arjona, S. Barakat, A. B. Sánchez, S. Segura, The rise of language models in mining software repositories: A survey, arXiv preprint arXiv:2604.00787 (2026).

[5] V. De Martino, J. Castaño, F. Palomba, X. Franch, S. Martínez-Fernández, A framework for using llms for repository mining studies in empirical software engineering, in: 2025 IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering (WSESE), IEEE, 2025, pp. 6–11.

[6] V. De Martino, S. Martínez-Fernández, F. Palomba, Green architectural tactics in ml-enabled systems: an llm-based repository mining study, Empirical Software Engineering 32 (2027) 1.

[7] D. Ma, S. Chen, Y. Yang, Y. Shi, Y. Yan, X. Gu, Llm agents can see code repositories, 2026. URL: https://arxiv.org/abs/2606.14061. arXiv:2606.14061.

[8] I. R. D. S. Simões, E. Venson, Evaluating source code quality with large language models: a comparative study, in: Proceedings of the XXIII brazilian symposium on software quality, 2024, pp. 103–113.

[9] Y. Zhang, M. Zhong, S. Ouyang, Y. Jiao, S. Zhou, L. Ding, J. Han, Automated mining of structured knowledge from text in the era of large language models, in: Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining, 2024, pp. 6644–6654.

[10] L. Radosky, I. Polasek, Large language models in software documentation and modeling: A literature review and findings, in: 2026 IEEE 24th World Symposium on Applied Machine Intelligence and Informatics (SAMI), IEEE, 2026, pp. 000263–000268.

[11] Q. Luo, Y. Ye, S. Liang, Z. Zhang, Y. Qin, Y. Lu, Y. Wu, X. Cong, Y. Lin, Y. Zhang, X. Che, Z. Liu, M. Sun, RepoAgent: An LLM-powered open-source framework for repository-level code documentation generation, in: D. I. Hernandez Farias, T. Hope, M. Li (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 436–464. URL: https://aclanthology.org/2024.emnlp-demo.46/. doi:10.18653/v1/2024.emnlp-demo.46.

[12] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al., Judging llm-as-a-judge with mt-bench and chatbot arena, Advances in neural information processing systems 36 (2023) 46595–46623.

[13] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical software engineering 14 (2009) 131–164.

[14] G. Rosa, J. J. Arias Rojas, J. M. González Barahona, Replication package, 2026. https://figshare.com/s/4d76bc8f792d39e75299.