Intelligent Analysis of Software Team Performance:   
An Agentic Approach

Vincil C. Bishop III1[[1]](#footnote-2)\*[0009-0009-2909-9685] and Lucas Krause2[0009-0004-5609-5472]

1 Colorado State University Alumnus, Ft. Collins, CO USA

Vincil.bishop24@alumni.colostate.edu

2 Colorado State University, Ft. Collins, CO USA  
[lucas.krause@colostate.edu](mailto:lucas.krause@colostate.edu)

**Abstract.** This research explores an agentic approach to analyzing software team performance through a combined quantitative and qualitative framework. Building on the Time-Delta Method for measuring software development contribution rates, this study introduces intelligent agentic mechanisms to identify critical areas of team performance requiring deeper analysis. Quantitative metrics will be evaluated using time delta and contribution rate imputation techniques, offering insights into team productivity and individual developer contributions over time.

In parallel, qualitative analysis will focus on team dynamics, collaboration, and workflow patterns to uncover root causes of performance issues. The agentic mechanisms will act as adaptive tools, guiding the investigation toward specific performance bottlenecks or strengths, enhancing the precision of the analysis. By merging these methodologies, the research aims to provide a holistic view of software team performance, offering actionable insights for improving team efficiency, collaboration, and overall project success.

**Keywords:** agentic, GenAI, LLM, performance, software quality, team

1. Introduction

This research explores whether an agentic approach can effectively analyze a repository and provide analysis comparable in quality to a comprehensive sequential analysis using Large Language Models (LLMs). The study investigates what advantages, if any, an agentic approach to performance analysis offers in terms of time/performance, quality/fidelity, and cost.

The exploration was designed to demonstrate techniques for analyzing large amounts of temporal data rather than definitively proving whether agentic analysis is superior. The deterministic nature of the underlying technologies makes precise comparison between results challenging. The primary goal was to demonstrate that large amounts of temporal performance data could be analyzed effectively, with insights derived through LLM-based methods.

While there are many interesting scientific aspects of LLMs and agentic performance that have been and could be researched, this exploration specifically focuses on demonstrating the utility of LLMs for analyzing engineering performance data, particularly software engineering performance data. The research compares sequential and agentic approaches to determine their relative strengths and limitations in this specific context.

1. Background and Related Work

The analysis of software team performance through repository data has evolved significantly with the emergence of Large Language Models (LLMs) and multi-agent systems. This research builds upon existing work in repository mining, LLM-based code analysis, and agent evaluation frameworks.

Repository-based team performance analysis has traditionally relied on quantitative metrics extracted from version control systems. Bock et al. [1] introduced tensor decomposition techniques for modeling group dynamics in open-source development, analyzing both communication patterns and version control data to understand developer interactions over time. Their mathematical framework complements the Time-Delta Method [2] by providing additional approaches for analyzing contribution patterns and team structures. While effective for capturing quantitative patterns, these traditional methods lack the capability to understand semantic context within code changes.

The advent of LLMs has enabled semantic analysis of entire repositories. Bairi et al. [3] developed CodePlan, a neuro-symbolic framework for repository-level code understanding that addresses the challenge of analyzing interdependent code across large codebases that exceed LLM prompt limits. Their incremental dependency analysis and change impact assessment demonstrate how LLMs can analyze complex repositories—a capability essential for evaluating team performance across interconnected codebases. Wang et al. [4] extended this with RLCoder, using reinforcement learning to improve repository-level code completion by learning which code components are most relevant for analysis, achieving 12.2% improvement in accuracy over previous methods.

Multi-agent frameworks represent an alternative approach to complex software analysis tasks. Wu et al. [5] introduced AutoGen, enabling customizable conversable agents that operate autonomously or with human intervention. Hong et al. [6] presented MetaGPT, which maps AI agents to actual software development team roles, simulating a software company's structure. These frameworks provide architectural patterns for creating specialized agents that could analyze different aspects of developer contributions, from commit patterns to code complexity metrics.

Evaluating agent-based systems requires robust metrics. Hey et al. [7] demonstrated the application of precision, recall, and F1 scores for assessing LLM-based systems in software engineering tasks through their NoRBERT framework for requirements classification. Their methodology for cross-project performance evaluation addresses generalization across diverse software projects—critical for analyzing teams with varying contribution patterns.

Recent work has questioned whether complex agent architectures are always superior. Xia et al. [8] compared sequential versus agentic approaches, demonstrating that their "Agentless" system outperformed complex agent-based methods on the SWE-bench Lite benchmark while reducing costs. This finding directly informs our research question about the relative merits of sequential versus agentic approaches for repository analysis.

While existing literature addresses individual components—repository mining, LLM-based analysis, multi-agent frameworks, and evaluation methodologies—no previous work combines these elements to comprehensively compare sequential and agentic approaches for software team performance analysis. Our research fills this gap by empirically evaluating both paradigms using the Time-Delta Method, providing insights into when each approach offers advantages for analyzing engineering performance data.

1. Experiment Design

To evaluate the efficacy and reliability of different analytical approaches for software team performance assessment, team will conduct a comprehensive series of experimental comparisons. The experimental framework will include three distinct evaluation paradigms designed to measure both the relative performance between methodologies and their internal consistency. The primary experiment will compare agentic analysis results against sequential analysis results to determine their relative effectiveness in generating meaningful insights from software repository data. This direct comparison will form the core of the investigation into whether agentic approaches offer advantages over more traditional sequential methods.

To establish a baseline understanding of methodological reliability, the researchers will conduct two additional determinism studies. The sequential determinism experiment will evaluate the consistency of the sequential approach by comparing outputs from multiple runs against each other, providing a measure of how stable and reproducible the sequential analysis paradigm is when applied to identical inputs. Similarly, the agentic determinism experiment will assess the consistency of the agentic approach by comparing multiple runs against each other, revealing the degree of variability inherent in more autonomous analytical systems.

For each experimental paradigm, the team will perform both quantitative evaluations using established metrics (accuracy, recall, precision, and F1 score) and qualitative assessments that examine the nature and content of the insights generated. This dual evaluation approach will allow researchers to characterize not only how often the different methodologies agree but also how they differ in their analytical focus, level of detail, and interpretive frameworks. The following sections present the detailed experimental design for each paradigm across all three repositories.

* 1. Experiment Dataset

For the dataset, commits from random contiguous time periods in 3 repositories were chosen. These repositories were selected because knowledge of the actual performance of the team could be verified from firsthand experience by the authors. This made it possible to sanity check the suitability of the sequential results as a control for the experiment.

***Table 1****. Summary of Repositories*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Repo | Total Commits | First Commit | Last Commit | Days |
| Repo1 | 2769 | 9/1/17 | 8/7/18 | 339 |
| Repo2 | 1211 | 7/7/21 | 7/18/23 | 740 |
| Repo3 | 371 | 4/14/20 | 2/9/21 | 301 |

For each repository in the dataset, various metrics were extracted and calculated. These metrics included:

* Commits per Contributor
* Commit Count
* Contribution Rate [9]
* Estimated Effort Hours Per Contributor [10]
* Estimated Effort Hours
* Mean Lines Delta Ratio
* Unique Contributors
* Word Change (Levenshtein Distance) per contributor.
* Word Change total.

The metrics were visualized using bar charts, correlative heat maps, as well as presented in tabular format. These items comprised the information fed into the large language model prompts for inference. Anthropic’s Claude Sonnet v3.7 was used for all experiments. [11]

A graph with numbers and a red line

Description automatically generated

**Fig 1**. LLM optimized bar chart example

A screenshot of a computer

Description automatically generated

**Fig 2**. LLM optimized heat map example

* 1. Experiment steps

To investigate software repository behavior and address specific analytical questions, two distinct methodologies were employed:

Sequential **Analysis (Control)**

The sequential analysis approach served as a baseline. This process involved:

1. **Data Filtering**: Filtering commit data relevant to the temporal periods specified in an object modeling the temporal question.
2. **Metrics Generation**: Generating descriptive statistics and metric summaries from this filtered data.
3. **LLM-Based Analysis**:
   * Selection of relevant commits based on the input data.
   * Inference of details and implications of changes within selected commits.
   * Synthesis of findings to answer the initial question.

The comprehensive analysis consisted of three levels:

1. **Macro Analysis**: Repository data summarized by months.
   * Questions about high and low performing periods were inferred.
2. **Micro Analysis**: For each high/low performing period from the Macro Analysis.
   * For high performing periods, commits and TDM metrics were analyzed to identify root causes.
   * For low performing periods, performance from adjacent periods was examined for contrast.
   * Answers to questions with root cause analysis were inferred.
   * Human verification of plausibility/accuracy ensured findings were rooted in repository metrics.
3. **Root Cause Analysis**: Final answers to Macro Analysis questions were inferred.
   * Human verification of plausibility/accuracy.
   * LLM evaluation according to a rubric. (LLM as Judge)
   * This output became the control of the experiment.

**Table 2**. Sample of macro level control questions.

|  |  |  |  |
| --- | --- | --- | --- |
| Repo | Question | Explanation | Analysis Type |
| Repo1 | Why did estimated effort hours and word change total spike dramatically in April 2018? | April 2018 shows a significant spike in both estimated effort hours (4041.7) and word change total (277,424), which are statistical outliers compared to other months. This represents a 68% increase in effort hours and a 68% increase in word changes from March 2018, followed by a dramatic 79% drop in May 2018. | high\_period |
| Repo1 | Why did May 2018 have the highest number of commits and commits per contributor despite having much lower effort hours? | May 2018 shows an interesting anomaly where it has the highest number of commits (487) and commits per contributor (18.0) of any month, yet the estimated effort hours dropped by 79% from April. This suggests a significant change in commit patterns or project phase. | high\_period |
| Repo2 | Why did the repository experience an extraordinary spike in effort hours and word changes during March-May 2022? | These three months show extremely high outlier values in effort hours, word changes, and per-contributor metrics compared to all other periods. The estimated effort hours peaked at 345.6 hours in March 2022, which is nearly 6 times higher than typical months. Similarly, word changes reached 75,128 in March, which is dramatically higher than the repository average. Understanding this intense development period could reveal important project milestones or major feature implementations. | high\_period |
| Repo2 | What caused the significant drop in development activity during September 2022? | September 2022 shows an anomalous drop in activity across multiple metrics. Estimated effort hours fell to just 5.2 hours (compared to 49.7 in August), commits dropped to 20 (from 83), and word changes decreased to 1,140 (from 10,811). This represents a sudden interruption in what was otherwise a period of consistent development, suggesting a possible project pause, team transition, or completion of a development phase. | relative\_period |
| Repo3 | Why did August 2020 show an extraordinary spike in effort hours and word changes compared to other months? | August 2020 stands out as a significant outlier with 1402.6 effort hours and 363,565 word changes, which is 3.4x higher than the next highest month. This suggests a major development push or project milestone. | high\_period |
| Repo3 | What caused the significant increase in development activity in January 2021 after several months of moderate activity? | January 2021 shows a notable increase in effort hours (391.9) and word changes (101,593) compared to the previous months, despite having fewer contributors (5) than previous months. This suggests a focused development effort by a smaller team. | high\_period |

Agentic Analysis (Subject)

Concurrently, an agentic approach was implemented using the CrewAI framework. This method utilized distinct software agents assigned specific, decomposed tasks:

* **Data Analyst Agent:** Gathering repository and author metrics.
* **Reporting Analyst Agent:** Processing and synthesizing findings.

These agents performed tasks including:

* Gathering metrics for both primary temporal periods and relative comparison periods.
* Identifying potentially significant commits.
* Obtaining detailed insights about those commits.
* Performing root cause analysis.

A manager agent coordinated these tasks, culminating in a final root cause analysis. The agentic framework was provided with the same tools used in the sequential analysis, with similar chain of thought instructions. The agent was tasked with answering the same Macro Analysis questions inferred by the experiment control, with the option to perform Macro/Micro analysis using tools as it deemed fit.

Experimental Comparison and Evaluation

1. A key aspect of this study design was the standardization of output format. Both methodologies produced their final analyses within a TemporalPeriodInsightContainer data structure, ensuring direct comparability.
2. To assess the quality of insights, an LLM was used for comparative judgment. For each research question, the LLM compared sequential and agentic outputs to identify and classify distinct insights into three groups:
3. Insights present only in the control analysis.
4. Insights present only in the subject (agentic) analysis.
5. Insights common to both analyses.
6. This categorization formed the basis for calculating performance metrics:

* True Positives (TP): Insights common to both analyses.
* False Positives (FP): Insights found only in the subject analysis.
* False Negatives (FN): Insights present in control but absent from subject analysis.

1. From these values, Analysis, Precision, Recall, and F1 were calculated.
2. To assess determinism, both the sequential and agentic processes were run against themselves, and the results compared for variance.
3. Experiment Results

The experimental evaluation encompassed three distinct analyses: the primary comparison between agentic and sequential approaches, and two determinism studies assessing the internal consistency of each method when run multiple times. These experiments yielded both quantitative metrics (accuracy, recall, precision, and F1 scores) and qualitative observations regarding analytical focus and insight generation across all three repositories. The following sections present these findings, highlighting performance differences, consistency patterns, and the characteristic strengths and limitations of each analytical approach.

* 1. Agentic vs. Sequential

Quantitative Results

**Table 3**. Agentic vs. Sequential performance results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Repo | Accuracy | Recall | Precision | f1 |
| Repo1 | 0.34 | 0.562 | 0.462 | 0.507 |
| Repo2 | 0.254 | 0.421 | 0.39 | 0.405 |
| Repo3 | 0.244 | 0.44 | 0.355 | 0.393 |

These metrics indicate that the agentic approach identified only about 40-56% of the insights found in the sequential analysis (recall), with moderate precision (36-46%) in the insights it did generate. Overall F1 scores ranged from approximately 0.39-0.51, suggesting substantial room for improvement in the agentic approach. [12, p. 157, 184, 203]

Qualitative Observations

Across all repositories, distinct analytical patterns emerged between sequential and agentic approaches that transcended quantitative metrics. Sequential analysis excelled in technical depth, code-level details, and precise measurement—a depth-first approach. Conversely, agentic analysis provided strategic breadth, business context, and holistic interpretation of development patterns. These consistent signatures across all repositories indicate inherent methodological differences rather than dataset-specific variations. Though quantitatively capturing fewer insights, the agentic approach contributed unique, complementary perspectives, suggesting optimal software team performance analysis might combine both methods—leveraging sequential precision with agentic contextualization.

Technical Specificity vs. Strategic Overview

Technical Specificity vs. Strategic Overview examines a fundamental difference between sequential and agentic analysis approaches: their orientation toward technical details versus strategic context. Sequential analysis excels at identifying specific code changes, architectural improvements, and quality metrics—providing precise diagnostic capabilities. Conversely, agentic analysis connects technical changes to broader project objectives and organizational goals, enabling higher-level planning. This distinction influences each approach's effectiveness for different stakeholders: engineering teams benefit from technical specificity, while product managers find strategic overview more actionable. The complementary nature of these perspectives suggests optimal analysis might incorporate elements of both approaches.

Sequential analysis demonstrated exceptional technical precision, revealing specific code-level changes that the agentic approach overlooked. For example, it identified critical Angular framework optimizations in Repository 1, including the replacement of ContentChild with ViewChild directives and improved component encapsulation techniques between July-August 2018. [12, p. 11] The sequential approach uniquely highlighted the mean lines delta ratio decrease from 0.3 in July to 0.2 in August 2018 [12, p. 138], revealing a transition toward code refinement rather than expansion—indicating the team shifted to bug fixing and cleanup after reaching a major project milestone. This technical insight provided a deeper understanding of Repository 1's development cycle, where the high-activity July period (with 33 unique contributors) [12, p. 1] was followed by a planned stabilization phase, context that was less apparent in the broader, more strategic agentic analysis.

Agentic analysis excelled at providing strategic context that sequential analysis often missed, particularly in Repository 2's March-May 2022 development spike. [12, p. 24] While sequential analysis identified technical implementations, the agentic approach revealed this represented a deliberate architectural transformation preparing the codebase for increased team scalability. In Repository 3's January 2021 activity surge, the agentic analysis uniquely categorized commit patterns (35% feature additions, 30% bug fixes, 20% refactoring, 10% documentation), revealing purposeful development methodology shifts. [12, p. 4] For Repository 1, it identified that despite team expansion from 25 to 33 contributors in July 2018, commits per contributor remained stable (13.6 to 13.0), indicating efficient onboarding processes. [12, p. 1] This strategic framing connected technical changes to broader organizational objectives that purely code-focused analysis overlooked.

Quantitative Precision vs. Contextual Interpretation

While precise metrics quantify observable phenomena objectively, contextual interpretation reveals underlying causes and strategic implications. This tension between "what happened" versus "why it matters" is critical—executives need contextual framing for decision-making, while engineering managers require exact measurements for process optimization.

Sequential analysis provided exceptional quantitative precision across all repositories. In Repository 1, it meticulously tracked the July-August 2018 transition, quantifying the 92% decrease in effort hours and the dramatic drop in per-contributor productivity from 52.7 to 7.4 hours. [12, p. 85, 139] For Repository 2, it precisely measured September 2022's activity decline with effort hours falling from 49.7 to just 5.2 hours (90% reduction) alongside an 83-to-20 commit count decrease. [12, p. 90] In Repository 3, it captured the extraordinary August 2020 spike (1402.6 hours, 363,565 word changes) and subsequent 97% effort reduction. [12, p. 79] These precise measurements enabled identification of statistically significant anomalies and cyclical patterns that would be missed with less granular analysis, providing objective evidence for development phase transitions.

Agentic analysis excelled at identifying micro-patterns and contextual interpretations missing from broad statistics. For Repository 1's April 2018 spike, it discovered 73.5% of all effort was concentrated in a single week (W14), revealing concentrated development rather than sustained activity. [12, p. 19] In Repository 3's August 2020 spike, it identified that two contributors were responsible for 67.1% of total effort, suggesting specialized expertise deployment rather than team-wide mobilization. [12, p. 18] The approach uniquely captured rapid development cycles in Repository 2's March 2022 peak, with 15 commits in a 2-hour window demonstrating coordinated sprints. [12, p. 4] These temporal insights connected raw metrics to development methodologies, revealing agile practices, pair programming sessions, and strategic resource allocation decisions that pure statistics obscured.

Team Dynamics and Human Factors

Team Dynamics and Human Factors explores how different analytical approaches assess the human dimension of software development. Understanding team composition, contributor patterns, and workload distribution provides critical insights beyond pure code metrics. This perspective is essential for sustainable team management—identifying burnout risks, resource allocation issues, and onboarding effectiveness that directly impact project success and team retention.

Sequential analysis excelled at identifying sustainability risks and objective team compositional changes. For Repository 1, it flagged potential burnout concerns during July 2018's peak when effort hours per contributor spiked to 52.7 hours—a 315% increase from June. [12, p. 138] In Repository 2, it detected concerning contributor concentration where the top contributor accounted for 37% of all commits during March-May 2022, creating key person dependencies. [12, p. 18] The approach methodically documented team composition fluctuations in Repository 3, tracking how the contributor count dropped 42% between August 2020 and September 2020 while maintaining strict objectivity about potential causes. [12, p. 102] This risk-focused assessment provided valuable early warnings about team health issues that could impact project sustainability.

Agentic analysis provided richer context about team dynamics and collaboration patterns. In Repository 1, it identified that despite the July 2018 team expansion from 25 to 33 contributors, the team maintained consistent productivity (13.0 commits per contributor), suggesting effective onboarding processes. [12, p. 1] For Repository 3's January 2021 activity surge, it recognized coordinated effort patterns where five contributors showed synchronized commit timing, indicating collaborative problem-solving sessions rather than independent work. In Repository 2's March 2022 spike, the agentic approach uniquely identified role-based contribution patterns—frontend specialists focusing on component architecture while backend developers simultaneously implemented API endpoints—revealing intentional parallelization strategies. These human-centered insights explained performance variations beyond what raw metrics alone could reveal. [12, p. 73]

Technical Explanations vs. Process Insights

Technical Explanations vs. Process Insights highlights how analytical approaches differ in their focus on what was changed versus how development proceeded. Technical explanations identify specific code modifications and architectural decisions, while process insights reveal development methodologies and efficiency patterns. This distinction is crucial because technical details inform implementation decisions, while process insights drive methodology improvements and team workflow optimization—both essential perspectives for comprehensive software engineering management.

Sequential analysis excelled at identifying concrete technical modifications that drove performance metrics. In Repository 2's March 2022 spike, it detected the removal of a large mock JSON file (1,331 lines) that significantly impacted metrics without representing actual feature development. [12, p. 155] For Repository 1's April 2018 activity surge, it identified specific API endpoint consolidation efforts and strict TypeScript interface implementation that improved type safety. In Repository 3's August 2020 peak, the approach discovered systematic data model normalization patterns, including consistent renaming conventions and schema standardization across 17 related files. [12, p. 79] These technical details revealed precisely what changed in the codebase—essential information for understanding how architectural evolution progressed and which specific technical decisions produced performance variations.

Agentic analysis provided deeper insights into development processes and methodologies. In Repository 1's May 2018 anomaly (highest commits despite lower effort hours), it identified a shift toward atomic commit practices suggesting test-driven development adoption. [12, p. 4] For Repository 3's January 2021 surge, it detected evidence of automated refactoring tool usage through consistent patterns across 42 files with identical formatting changes. [12, p. 58] In Repository 2's June 2023 activity, the agentic approach uniquely recognized a comprehensive quality assurance initiative with parallel development of unit tests, integration tests, and documentation—suggesting a quality-focused development culture shift. [12, p. 35] These process insights revealed how development was conducted rather than just what was built, offering valuable perspectives on methodology effectiveness, team maturity evolution, and development practice adoption that technical explanations alone couldn't provide.

Consistency in Findings

Both sequential and agentic approaches consistently identified core development patterns across all repositories. In Repository 1, both recognized the July 2018 peak as representing a critical project milestone with coordinated team effort, followed by an intentional phase transition in August. [12, p. 2] For Repository 2's September 2022 activity drop, both approaches correctly identified it as representing project completion rather than productivity issues. In Repository 3's August 2020 spike, both analyses recognized the architectural refactoring nature of changes and their purpose in enabling future feature development. [12, p. 50] This consistent identification of major development anomalies, project phase transitions, and architectural improvement initiatives across both approaches provides strong validation of these findings. Where the approaches differed was primarily in analytical depth versus breadth rather than fundamental disagreement, suggesting they offer complementary perspectives rather than contradictory interpretations.

* 1. Sequential vs. Sequential (Determinism Analysis)

Determinism analysis is crucial when utilizing inherently non-deterministic LLMs for critical analysis tasks. By comparing sequential analyses against themselves, we establish a baseline for methodological reliability, quantifying the expected variability even in structured approaches. This benchmark enables proper interpretation of differences between methodologies and informs confidence levels in analytical conclusions derived from LLM-based systems.

Quantitative Results

**Table 4**. Sequential vs. Sequential performance results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Repo | Accuracy | Recall | Precision | f1 |
| Repo1 | 0.651 | 0.875 | 0.718 | 0.789 |
| Repo2 | 0.647 | 0.846 | 0.733 | 0.786 |
| Repo3 | 0.594 | 0.76 | 0.731 | 0.745 |

The sequential approach demonstrated higher consistency when compared against itself, with recall values of 76-88% and precision values of 72-73%. This suggests that even the sequential process has some inherent variability, but significantly less than the agentic approach. [12, p. 377, 398, 413]

Qualitative Observations

The sequential analysis demonstrated remarkable consistency across all repositories when compared against itself. Multiple execution runs produced highly similar results, with the sequential approach identifying the same core insights with high reliability. When variations did occur, they manifested primarily in subtle differences of emphasis rather than substantive disagreements in findings or conclusions. Quantitative metrics remained nearly identical across runs, exhibiting minimal statistical variance. Both sequential analyses consistently reached equivalent conclusions regarding root causes of performance patterns and raised similar follow-up questions about potential external factors. This high degree of agreement in both analytical outcomes and quantitative measurements suggests that the sequential approach offers a stable, deterministic methodology for software team performance analysis, providing reproducible results that can serve as a reliable foundation for decision-making processes.

* 1. **Agentic vs. Agentic (Determinism Analysis)**

Evaluating the consistency of agentic approaches reveals the inherent trade-offs between autonomous decision-making and analytical stability in LLM systems. This analysis quantifies variability introduced by the agentic framework itself, establishing critical boundaries of reliability when employing such methods for performance analysis and decision support in engineering contexts.

### Quantitative Results

**Table 5**. Agentic vs. Agentic performance results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Repo | Accuracy | Recall | Precision | f1 |
| Repo1 | 0.414 | 0.615 | 0.558 | 0.585 |
| Repo2 | 0.356 | 0.618 | 0.457 | 0.525 |
| Repo3 | 0.413 | 0.633 | 0.543 | 0.585 |

The agentic approach showed moderate consistency when compared against itself, with recall values of 62-63% and precision values of 46-56%. This indicates more variability in the agentic process compared to the sequential approach. [12, p. 22, 47, 65]

Qualitative Observations

When evaluating self-consistency, agentic analyses demonstrated notable patterns of both agreement and divergence across repositories. While consistently identifying overarching phenomena and primary causal factors, agentic analyses exhibited substantial variability in their emphasis, technical focus, and quantitative details. Both control and subject analyses generally converged on similar high-level conclusions about team performance patterns and development trajectories. However, they often diverged significantly in their selection of supporting evidence, specific technical aspects highlighted, and granular metrics cited. This marked contrast with the sequential approach's higher consistency suggests that agentic frameworks introduce additional variability through their autonomous selection processes and decision-making pathways. The distributed cognition inherent in multi-agent systems appears to amplify the non-deterministic qualities of the underlying LLM technologies, creating a more exploratory but less predictable analytical approach.

1. Analysis & Discussion
   1. Factors Affecting Agentic Performance

Upon reviewing traces from the agentic LLM transactions, the tendency for the agentic process to offer broader and more strategic interpretations was identified as stemming from two primary factors:

Commit Selection Differences

At a fundamental level, the agentic process had different root cause commits to examine. This was because the LLM was given the freedom to choose which commits it thought most affected the condition it was presented with. The agentic process often chose different commits than the sequential process, which caused the underlying data the agentic process had available to infer an explanation to be different.

This autonomy in commit selection represents both a strength and weakness of the agentic approach:

* Strength: The ability to independently identify potentially relevant commits allows for novel insights that might be missed in a more structured approach.
* Weakness: Without clear guidance on commit selection criteria, the agentic process may focus on less relevant commits, leading to interpretations that miss key technical details.

Information Prioritization Differences

The agentic process was given latitude to prioritize all information it had, and in some cases, the explanation of actual commit actions competed with other information in context for root-cause prioritization. Other information often drew on general knowledge about software engineering team performance from the LLM's training, so inferred analysis frequently deviated from focused insights derived from the repository itself.

This tendency manifested in several ways:

* Agentic analyses provided more strategic context but sometimes at the expense of technical specificity.
* General software engineering principles were sometimes applied without sufficient grounding in the specific repository data.
* Broader interpretations were offered that connected to business objectives and team dynamics, even when direct evidence for these connections was limited.
  1. Opportunities for Improvement

There exist several opportunities to improve the performance of the agentic process:

Agentic Framework Optimizations

Potential improvements include:

* Moving to a different agentic framework like LangGraph, Pydantic.ai, or implementing a custom agentic workflow.
* Addressing the "black box" nature of interactions with the Crew.AI framework
* Creating more transparent agent-to-agent communication patterns.
* Implementing better monitoring and logging of agent reasoning processes.

These changes would come with tradeoffs, including increased implementation complexity among others.

Prompt Optimization

Based on observed deviations from the sequential workflow, specific optimizations could be introduced:

* Enhanced prompts that better focus the agentic process on the types of information desired in the final output.
* Inclusion of examples for model output to guide the analysis toward more technically grounded interpretations.
* Adjustments to the configuration of the number and type of agents employed to elicit analysis.
* More explicit instructions regarding the balance between technical specificity and strategic context.

Hybrid Approach Development

Given the complementary strengths of sequential and agentic approaches, developing a hybrid methodology could leverage the advantages of both:

* Using sequential analysis for technical specificity and precise quantitative analysis.
* Employing agentic analysis for strategic context and broader interpretations.
* Creating a framework that synthesizes insights from both approaches.
* Implementing validation mechanisms to ensure technical accuracy while maintaining strategic relevance.

1. Conclusions
   1. Summary of Findings

This research provides several key insights into the use of sequential versus agentic approaches for software team performance analysis:

Effectiveness of Visual Data Summarization

Summarizing large amounts of data visually in a format optimized for interpretation by an LLM, and providing those visualizations in a multimodal fashion, proved to be an effective way to analyze large datasets. This approach worked particularly well where tabular data could otherwise overwhelm the LLM's context and exceed the model's ability to reason over large amounts of textual information.

Using this technique in stages—first examining large amounts of information, then focusing on smaller, more impactful areas for analysis—was an effective way to approach intelligent analysis of large amounts of data, potentially augmenting or accelerating human analysis of engineering performance data.

Comparative Advantages

The comparative analysis revealed that neither agentic nor sequential approaches to generative AI analysis are universally superior. Sequential methodologies demonstrated comparable or superior effectiveness for many engineering analysis tasks, challenging the assumption that autonomous agent systems inherently provide better analytical outcomes. This finding has significant implications for selecting appropriate AI architectures in software engineering contexts.

The sequential approach showed several advantages:

Sequential approaches exhibited superior determinism with consistency values reaching 76-88% across repositories, providing reproducible results essential for engineering applications requiring auditable decisions. These methods demonstrated enhanced technical precision, directly linking observations to specific commits and code-level evidence. The structured workflow produced quantitative analyses with minimal statistical variance, closely aligning with expert engineer expectations. This deterministic behavior makes sequential approaches particularly valuable for safety-critical systems, regulatory compliance, and situations demanding traceable analytical pathways from data to conclusions.

Agentic systems demonstrated complementary strengths in contextual synthesis and exploratory analysis. These approaches excelled at connecting technical changes to strategic business objectives, identifying team dynamics patterns, and exploring non-obvious causal relationships across development cycles. The autonomous decision-making capability enabled flexible hypothesis exploration, uncovering insights about organizational factors and development methodologies often overlooked by rigid sequential processes. This broader interpretive capacity proves valuable for strategic planning, team optimization, and understanding complex sociotechnical systems where human factors significantly influence engineering outcomes.

* 1. Research Contributions

This research advances engineering science through novel methodological frameworks and empirical insights into LLM analytical capabilities. The work establishes foundational techniques for applying generative AI to temporal engineering data analysis while revealing fundamental characteristics of sequential versus agentic approaches.

Methodological Innovations

The methodological innovations center on integrating quantitative metrics with qualitative analysis through a structured LLM framework. The Time-Delta Method metrics—including effort hours, contribution rate, and code change patterns—were synthesized with visual representations optimized for LLM interpretation. This multimodal approach enabled processing of complex temporal datasets that exceed traditional tabular analysis limits. The standardized comparison methodology introduced reproducible evaluation criteria using LLM-as-Judge techniques, establishing benchmarks for assessing analytical approach effectiveness. These innovations demonstrate how engineering performance data can be systematically analyzed at macro and micro levels through staged LLM processing.

Insights into LLM Analysis Capabilities

The research revealed critical insights about LLM analytical behaviors that inform engineering applications. Sequential approaches demonstrated superior performance in technical specificity, achieving 72-73% precision in identifying code-level changes and architectural decisions. Conversely, agentic systems excelled at strategic synthesis, connecting technical modifications to organizational objectives with 40-60% broader contextual coverage. These findings exposed fundamental trade-offs between deterministic precision and exploratory breadth, suggesting optimal performance analysis requires hybrid approaches. The identified patterns in LLM interpretation—particularly the tendency for agentic systems to prioritize general software engineering principles over repository-specific evidence—provide crucial guidance for designing effective AI-assisted engineering analysis systems.

* 1. Broader Applications

An agentic approach should be considered when inputs are unpredictable, such as in a chat scenario with a human in the loop. When a workflow is predictable and can be addressed with conditional instead of intelligent agentic processes, a sequential approach may offer advantages including more deterministic outputs, ease of maintenance, debugging, and explain ability.

The techniques developed in this research can be applied to other performance analysis datasets, especially those of a temporal nature where root cause analysis should be performed for larger macro-level observations. Potential applications include:

* Manufacturing process optimization
* Supply chain performance analysis
* Financial system performance evaluation
* Infrastructure reliability assessment
* Service delivery efficiency analysis
* Healthcare process improvement
* Educational system performance evaluation
  1. Future Work

This research establishes a foundation for advancing LLM-based engineering analysis, with opportunities spanning framework development, validation extensions, and methodological refinements that promise enhanced analytical capabilities across engineering domains.

Framework Development

Future framework development should focus on hybrid sequential-agentic architectures that leverage the complementary strengths identified in this study. Implementation of transparent agentic systems with explainable decision pathways would address the "black box" limitations observed in current approaches. Specialized agents could be developed for specific analytical tasks—commit selection, metric calculation, and strategic synthesis—while integration with advanced visualization techniques would enhance multimodal data processing capabilities.

Validation and Extension

Validation efforts must extend beyond software repositories to encompass diverse engineering contexts including manufacturing processes, infrastructure systems, and supply chain networks. Comparative studies with human expert analyses would establish performance baselines and identify areas where AI assistance provides maximum value. Development of standardized benchmarks across engineering domains would enable systematic evaluation of analytical approaches, fostering reproducible research and practical implementation guidelines for industry adoption.

Methodological Refinements

Methodological refinements should prioritize prompt engineering techniques that balance technical precision with strategic insight. Enhanced evaluation metrics beyond traditional accuracy measures could capture the nuanced value of different analytical approaches. Investigation of determinism factors across various LLM architectures would inform reliability requirements for critical engineering applications. Techniques for systematically combining multiple methodologies—potentially through ensemble approaches or meta-analysis frameworks—represent crucial areas for improving analytical comprehensiveness and accuracy.

This research provides a foundation for further exploration of both sequential and agentic approaches to engineering performance analysis, with potential applications across a wide range of domains beyond software development.

# Acknowledgements

The authors would like to express their deepest gratitude to the Systems Engineering Department at Colorado State University, Fort Collins, for their invaluable support in making this research possible.

During the preparation of this work, artificial intelligence was used in the research and writing processes. After using these services, the content was reviewed and edited as needed and full responsibility is taken by the authors for its final form.

References

[1] T. Bock, A. Schmid, and S. Apel, "Measuring and Modeling Group Dynamics in Open-Source Software Development: A Tensor Decomposition Approach," *ACM Transactions on Software Engineering and Methodology,* vol. 31, no. 2, p. 19, 2021/02 2021, doi: 10.1145/3473139.

[2] V. Bishop Iii, S. J. Simske, M. Vans, Y. Malaiya, and I. Ray, "Time-delta method for measuring software development contribution rates," Mountain Scholar : Digital Collections of Colorado, 2024. [Online]. Available: <https://hdl.handle.net/10217/239202>

[3] R. Bairi *et al.*, "CodePlan: Repository-level Coding using LLMs and Planning," in *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2024, doi: 10.1145/3643778.3643783. [Online]. Available: <https://doi.org/10.1145/3643778.3643783>

[4] Y. Wang *et al.*, "RLCoder: Reinforcement Learning for Repository-Level Code Completion," in *Proceedings of the International Conference on Software Engineering (ICSE 2025)*, 2024.

[5] Q. Wu *et al.*, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework," *arXiv preprint,* 2023, doi: 10.48550/arXiv.2308.08155.

[6] B. Hong *et al.*, "MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework," *arXiv preprint,* 2023, doi: 10.48550/arXiv.2308.00352.

[7] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "NoRBERT: Transfer Learning for Requirements Classification," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020, pp. 169-179, doi: 10.1109/RE48521.2020.00028. [Online]. Available: <https://ieeexplore.ieee.org/document/9218141/>

[8] C. S. Xia, Y. Deng, S. Dunn, and L. Zhang, "Agentless: Demystifying LLM-based Software Engineering Agents," in *Proceedings of 33rd ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2025)*, 2024.

[9] V. Bishop, III, "Model Contribution Rate Theory: An Empirical Examination," pp. 1-1, 2024/12/08 2024, doi: 10.48550/arXiv.2412.05978.

[10] V. Bishop, III and S. Simske, "Contribution Rate Imputation Theory: A Conceptual Model," *arXiv,* vol. 2410.09285, p. 14, 2024, doi: 10.48550/arXiv.2410.09285.

[11] *Claude 3.7 Sonnet*. (2025). Anthropic. [Online]. Available: <https://www.anthropic.com>

[12] V. Bishop Iii. *Experiment Output Data: Intelligent Analysis of Software Team Performance*, ResearchGate, doi: 10.13140/RG.2.2.15235.00802.

1. \* Not related to work at Amazon, Inc. [↑](#footnote-ref-2)