Beyond Detection: A Multi-Agent Framework for Root Cause Analysis of Financial Discrepancies in Distributed Environments

Dane Thomas
Amazon
danent@amazon.com

Daksha Yadav
Amazon
dakyadav@amazon.com

Boyang Tom Jin
Amazon
boyanjin@amazon.com

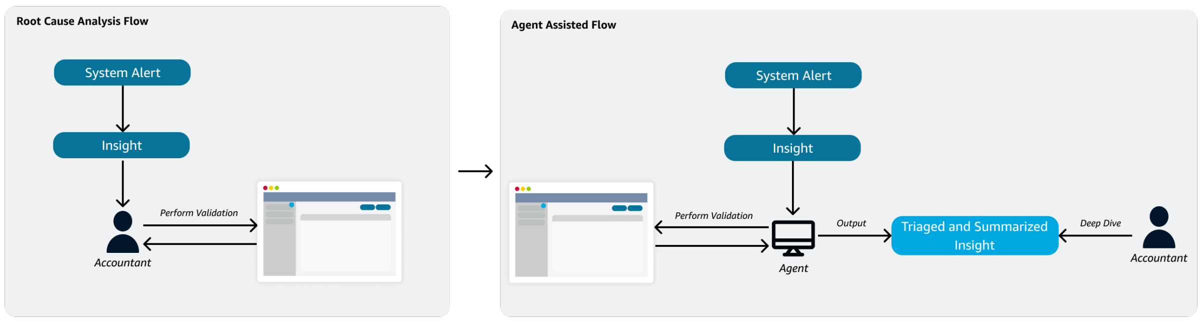


Figure 1: A comparison of the existing root cause analysis workflow with the proposed agent assistant workflow.

Abstract

The increasing complexity and fragmentation of financial systems in large organizations have created significant challenges for financial teams, particularly in performing real-time, end-to-end validation, as existing validation methods relying on static rules or batch processing are often inadequate for today's dynamic financial environments. This paper introduces a novel approach using Large Language Model (LLM)-based browser agents within a multi-agent framework to enhance financial validation processes. The framework leverages domain-specific agents that autonomously navigate web-based financial platforms to validate data, interpret discrepancies, and perform root cause analysis, ensuring higher accuracy, transparency, and auditability compared to traditional systems. A synthetic dataset and controlled simulation environment were used to evaluate the framework's performance across 20 distinct financial scenarios, revealing significant improvements in validation accuracy (from 40% with a Vanilla agent to 65% with the proposed approach). The results indicate that the proposed multi-agent approach, by isolating validation tasks into specialized agents and orchestrating a coordinated investigation, provides a more reliable, scalable, and interpretable solution for high-stakes financial environments.

CCS Concepts

• **Computing methodologies** → **Machine learning**.

Keywords

large language models, agents, machine learning, web use agents, financial assistant

ACM Reference Format:
Dane Thomas, Daksha Yadav, and Boyang Tom Jin. 2025. Beyond Detection: A Multi-Agent Framework for Root Cause Analysis of Financial Discrepancies in Distributed Environments. In *Proceedings of KDD 2025 Workshop on Agentic AI for Enterprise (KDD'25).* Toronto, ON, Canada, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Today's financial landscape, especially within large organizations, is defined by a growing and critical challenge: the inability to perform real-time, end-to-end validation across a fragmented and complex ecosystem of financial systems. This fragmentation arises from the widespread use of diverse technologies, including legacy mainframes, on-premise ERPs, cloud-based SaaS platforms, and API-driven third-party tools. The result is a web of disconnected data silos that severely hinder finance and audit teams from maintaining financial integrity. In such environments, validation plays a crucial role, not only in ensuring that anomalies are detected, but in confirming that insights are accurate, explainable, and actionable across multiple systems. Without robust validation, financial teams risk acting on false positives, missing hidden discrepancies, or failing to trace anomalies back to their root causes, all of which can have serious operational consequences.

Existing validation methods, which typically rely heavily on static rules or batch processing, are inadequate for today's dynamic, high-velocity financial environments. These outdated approaches lead to delayed anomaly detection, shallow or incomplete root cause analysis, and greater exposure to financial risk. The consequences are far-reaching: missed fraud, unaddressed material misstatements can result in substantial financial losses, reputational damage, and costly penalties. Despite technological advances, many enterprises still depend on manual reconciliation to bridge systemic gaps, an approach that is resource-intensive, error-prone, and ill-suited to modern demands. As long as data remains siloed and real-time validation remains elusive, systemic vulnerabilities will persist, growing unchecked beneath the surface.

To address these challenges, organizations require more than just automation. They need intelligent, adaptive systems capable of reasoning, context-switching, and collaborating across heterogeneous environments. This is where Large Language Model (LLM)-based agents emerge as a transformative solution. Unlike traditional rule-based bots, LLM agents can interpret unstructured financial records, navigate disparate data sources, and engage in multi-turn interactions to synthesize complex information [2]. Their ability to understand natural language, infer missing context, and generate human-like insights enables them to go beyond anomaly detection and contribute to root cause analysis in real time.

A particularly promising application of LLMs in financial operations lies in the development of LLM-based browser agents [1, 3], autonomous agents that interact with web-based financial platforms in a human-like manner. These agents can extract reports, interpret dashboards, follow audit trails, and cross-reference information, all through a browser interface, without requiring direct API integration. In fragmented environments where critical data is locked behind proprietary interfaces, these agents serve as intelligent surrogates for human users, capable of adapting to changing UIs and delivering structured, explainable outputs.

In this paper, a multi-agent framework is proposed that employs LLM-based browser agents, each specialized in a specific accounting domain such as General Ledger (GL), Sub-Ledger (SL), and others. Unlike anomaly detection systems that surface statistical outliers or rule-based exceptions, the focus here is on post-detection validation, a critical phase in high-stakes financial environments. The framework consumes alerts or insights from upstream systems (e.g., reconciliation engines, BI dashboards, custom detectors) and orchestrates a coordinated investigation across otherwise disconnected platforms. This emulates the workflow of a forensic accounting team but offers higher speed, consistency, and auditability.

Each domain-specific agent operates autonomously within a browser environment, using LLM capabilities to navigate financial systems, locate supporting evidence (e.g., journal entries, invoice trails, tax filings), and validate key attributes such as transaction amounts, dates, and GL codes. These agents apply contextual reasoning to identify causes of discrepancies, ranging from timing mismatches and posting errors to inter-system reconciliation failures. Their structured findings are passed to a central Supervisor Agent, which performs cross-agent synthesis, applies meta-level validation logic, and compiles a comprehensive audit trail that includes root cause narratives and confidence scores.

This layered architecture is particularly important in enterprise finance where correctness, transparency and traceability are desired. Root cause analysis must go beyond numeric deviation to incorporate semantic understanding of financial workflows, system-specific logic, and evolving compliance requirements. Moreover, auditability demands that each validation step be reproducible and interpretable by human reviewers. By enabling browser-native, domain-aware reasoning without deep integration or centralized data pipelines, the proposed system bridges both technical and organizational silos, delivering explainable, resilient validation across complex financial landscapes.

As financial operations grow more complex and distributed, the case for intelligent, autonomous agents becomes imperative. LLM-based agents represent a paradigm shift, moving from passive validation to active investigation, from isolated rule engines to collaborative multi-agent systems. In the sections that follow, we describe the architecture and implementation of this system and evaluate its performance in realistic enterprise scenarios.

2 Related Works

Recent advancements in LLMs have catalyzed the development of autonomous web agents capable of performing complex tasks across diverse online environments. A significant contribution in this domain is AgentOccam, which emphasizes the importance of aligning an agent's observation and action spaces with the LLM's inherent capabilities. By refining these interfaces, AgentOccam achieved notable performance improvements on the WebArena benchmark, surpassing previous methods without relying on in-context examples or elaborate prompting strategies [8].

The WebArena environment [9] itself represents a leap forward in creating realistic and reproducible settings for evaluating web agents. It encompasses fully functional websites across domains such as e-commerce, social forums, collaborative software development, and content management. Despite these advancements, evaluations within WebArena revealed that even state-of-the-art agents like those powered by GPT-4 achieved only a 14.41% success rate on complex tasks, highlighting the challenges that remain in this field.

In the realm of user experience (UX) design, UXAgent [4] introduces an innovative framework for automated usability testing. Leveraging LLM agents, UXAgent simulates user interactions with web interfaces, enabling UX researchers to conduct large-scale testing without human participants. This approach facilitates the collection of both qualitative and quantitative data, streamlining the iterative design process .

Benchmarking the practical capabilities of LLM agents in real-world tasks, TheAgentCompany [7] provides an extensible framework that simulates a professional work environment. Within this setting, agents perform tasks such as browsing the web, coding, and collaborating with simulated coworkers. Findings indicate that while agents can autonomously complete simpler tasks, complex, long-horizon tasks remain challenging, with the best-performing agents achieving a 24% completion rate .

Addressing the dynamic nature of web environments, WebCanvas [6] offers a novel evaluation framework that captures the evolving aspects of web interactions. It introduces the Mind2Web-Live dataset, comprising 542 tasks with 2,439 intermediate evaluation states, and provides tools for continuous benchmarking of web agents. This framework emphasizes the need for adaptability in agents to maintain performance amidst the ever-changing web landscape.

While these works reflect growing sophistication in LLM-driven web agency, none explore multi-agent coordination or domain-specific validation in high-stakes contexts such as finance and accounting. Existing agents primarily focus on web navigation, data extraction, or UI interaction, but do not model the cross-system reasoning or traceability required in enterprise financial operations. In particular, the use of browser-native LLM agents for post-anomaly validation, coordinating across fragmented systems to replicate how finance teams conduct forensic investigations, remains an unexplored direction. This gap is significant given the complexity and compliance requirements of financial domains, where interpretability and auditability are as critical as accuracy.

3 Synthetic Dataset Generation and Simulation Environment

In this research, a synthetic dataset within a controlled simulation environment is generated that replicates the complex, interconnected systems of a large organization's financial ecosystem. This simulated environment is designed to reflect the challenges faced in modern enterprise accounting, where multiple distributed systems and diverse financial operations generate data that require validation and reconciliation.

3.1 Simulation Environment

In this simulation, series of interconnected web-based platforms representing key financial domains and business processes within an enterprise are created. The following systems are simulated to reflect the core functionalities typically found in a large-scale accounting environment:

- General Ledger (GL): The central accounting record for all financial transactions, ensuring consistency across all subsidiary systems.
- Subledger (SL): Contains detailed transaction records for specific accounts such as accounts payable, accounts receivable, and inventory management.
- Ordering Systems: Manages customer orders, tracking of goods, and payment status.
- Shipment Systems: Tracks the shipment and delivery status of ordered goods and services.
- Payment Processing Systems: Manages inbound and outbound payments, including bank transfers, credit card transactions, and digital payments.
- Vendor Management Systems: Tracks vendor contracts, supplier invoices, and purchase orders.
- Invoice Processing Systems: Handles the creation, validation, and tracking of invoices for products or services rendered.
- Payroll Systems: Manages employee salary and benefit processing, including tax calculations and statutory deductions.

These simulated systems are integrated to allow cross-system dependencies, where data from one system (e.g., a payment transaction) may trigger or impact another (e.g., an invoice or shipment update). The interactions between these systems create an environment with realistic, dynamic data flows, akin to those found in large organizations that rely on multiple interconnected systems for daily operations.

3.2 Scenario Generation

We synthetically generated a series of complex, multi-dimensional scenarios, each involving anomalies or inconsistencies within the simulated environment. These scenarios were designed to reflect a wide range of common accounting issues that could arise in real-world settings and require validation by an accounting expert. Each scenario consists of a set of insights, anomalies, or discrepancies that need to be validated, cross-referenced, and explained. The following types of synthetic scenarios are generated:

- Cross-System Inconsistencies: Discrepancies between financial systems, such as mismatches between GL and SL entries, missing or erroneous payment information, or unaccounted-for inventory movements.
- Data Entry Errors: Incorrect or incomplete data entries (e.g., incorrect vendor details, payment amounts, or payroll discrepancies) that require rectification.
- Fraud Detection Scenarios: Anomalous behavior suggesting potential fraud (e.g., duplicate invoices, irregular employee payroll, or unusual payment patterns).
- Data Delays: Data inconsistencies due to delays between connected systems (e.g, system outage delaying payment processing or delayed customer payments).

These scenarios are designed to simulate real-world issues that would require the expertise of an accounting professional to validate, interpret, and resolve. Each scenario involves interactions between multiple systems, resulting in complex chains of data that need to be analyzed and cross-validated. In total 20 such scenarios are generated, comprising 12 Valid Insights and 8 False Alarms. The full list of scenarios, including system interactions and expected validation logic, is provided in Appendix B.

4 Proposed Approach

This work proposes a modular, multi-agent framework powered by LLM-based browser agents to assist accounting and finance teams in performing root cause analysis of financial discrepancies across distributed and siloed systems. Unlike traditional anomaly detection tools, which focus on identifying deviations using rules or statistical models, the proposed system operates downstream, after anomalies have been flagged, focusing instead on validation, contextual interpretation, and explainable root cause tracing. This approach is particularly critical in modern enterprise settings where financial data spans heterogeneous systems such as legacy ERPs, cloud-based SaaS platforms, and third-party vendor portals, many of which lack robust API access and require human-like interaction via web interfaces.

As shown in Figure 2, the architecture comprises a set of LLM-enhanced browser agents, each specialized in a specific financial domain. These agents are capable of navigating complex enterprise web applications, extracting structured and unstructured data, interpreting domain-specific semantics, and applying accounting logic in real-time. They operate autonomously within secure browser environments, mimicking the behavior of expert human analysts. Examples of such agents include:

- GL Agent: Validates journal entries, checks consistency in account balances, and ensures that high-level transactions

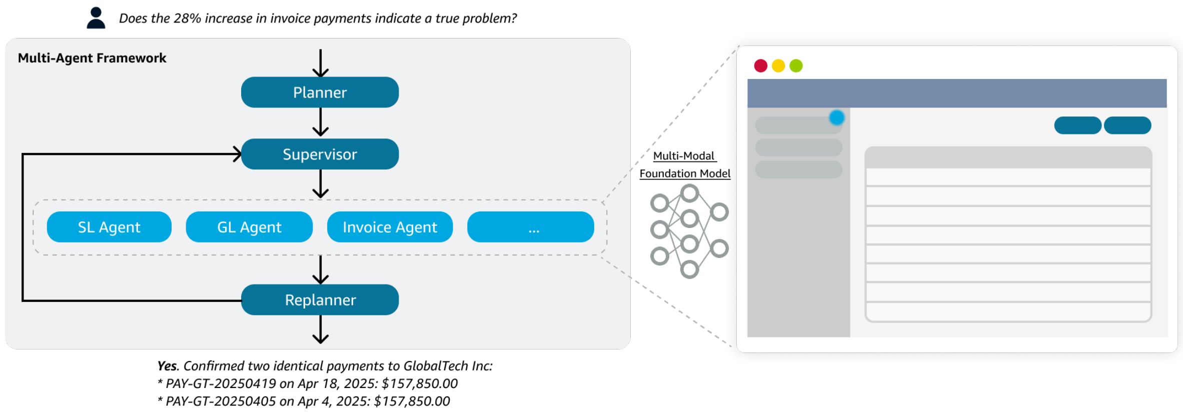

Does the 28% increase in invoice payments indicate a true problem?
Multi-Agent Framework
Planner
Supervisor
SL Agent
GL Agent
Invoice Agent
...
Replanner
Multi-Modal Foundation Model
Yes. Confirmed two identical payments to GlobalTech Inc:
* PAY-GT-20250419 on Apr 18, 2025: $157,850.00
* PAY-GT-20250405 on Apr 4, 2025: $157,850.00

Figure 2: Overview of the proposed multi-agent framework for automated financial validation. The system comprises a Planner that generates the validation workflow, a Supervisor that delegates tasks to specialized agents (e.g., SL Agent, GL Agent, Invoice Agent), and a Replanner that handles task reassignments or failures. Each agent interacts with simulated enterprise systems via an LLM-powered browser interface to retrieve and reason over financial data.

recorded in the General Ledger align with underlying subledger entries. It flags anomalies such as duplicate postings, missing entries, or misclassified expenses.

- SL Agent: Focuses on transactional detail within Subledgers, such as accounts payable, accounts receivable, and inventory adjustments. It ensures that entries are correctly posted, reconciles them with the GL, and identifies timing mismatches or data integrity issues.
- Invoice Agent: Navigates invoice processing systems to validate invoice generation, approval workflows, and payment status. It cross-references invoices with purchase orders, delivery records, and payment confirmations to detect inconsistencies, such as phantom invoices or double billing.
- Vendor Agent: Reviews vendor contracts, purchase orders, and associated invoices. It validates contract terms against billing behavior, ensures vendor master data integrity, and flags duplicate vendors, suspicious contract amendments, or mismatched invoice amounts.
- Order Agent: Operates within customer ordering systems to verify order integrity, customer fulfillment timelines, and revenue recognition compliance. It detects incomplete order-to-cash cycles or inconsistencies between order entries and downstream financial records.
- Shipment Agent: Validates the linkage between order fulfillment and logistics systems by checking whether goods/services were shipped according to contractual terms and verifying that shipment status aligns with billing and delivery records.
- Payment Agent: Monitors inbound and outbound payments to ensure completeness and accuracy. It validates payment references, matches them to invoices or payroll records, and detects anomalies such as payment duplication, routing errors, or policy violations.
- Payroll Agent: Navigates payroll systems to verify salary disbursements, tax calculations, and benefits deductions. It ensures compliance with internal policies, and flags anomalies such as overpayments, classification errors, or unapproved bonuses.

At the top of the framework is a Planner Agent responsible for laying out the overall workflow. Upon receiving a system-generated anomaly or insight (e.g., an invoice mismatch, unreconciled journal entry, or duplicate payment), the planner decomposes the problem and delegates investigative sub-tasks to relevant domain agents. This plan is received by the Supervisor Agent which orchestrates the execution of tasks by the system expert Agents until an answer is determined. Each agent performs a focused analysis, querying appropriate systems, validating data consistency, and returning a structured response consisting of a verdict, supporting evidence, and a natural language explanation. Each response is passed to a Replanner Agent, which updates the shared agent memory with logs of actions taken and insights gathered, maintaining a transparent trace of the workflow. This updated state is then returned to the Supervisor to inform subsequent task execution.

The Supervisor Agent performs three core functions upon receiving domain-level inputs: (1) *cross-agent validation*, ensuring coherence across responses and resolving conflicting interpretations; (2) *root cause synthesis*, assembling a traceable, explainable narrative of the discrepancy; and (3) *audit trail generation*, compiling a structured report including system interactions, extracted data, validation steps, and justifications.

By decomposing the validation process into modular, domain-specific tasks and leveraging browser-based automation for system interoperability, the proposed framework delivers significantly improved accuracy, transparency, and adaptability in performing financial root cause analysis.

5 Experimental Evaluation

5.1 Experimental Setup

To rigorously evaluate the performance of the proposed multi-agent framework, 20 distinct validation scenarios were designed (outlined in Section 3), each executed five times, resulting in 100 total runs. In every run, the agent classifies an alert as "Valid Insight," "False Alarm," or "Needs Manual Review" (the latter treated as valid, since it defers judgment to a human expert). This setup thus reduces to a binary decision problem: insight versus non-insight, while preserving a safety valve for low-confidence cases. This experimental setup serves two critical purposes: first, it assesses the system's accuracy in correctly validating insights across diverse financial conditions, and second, it measures consistency and repeatability, which are essential for ensuring auditability in enterprise financial systems. Performance was evaluated using the metrics outlined in the following sections.

5.2 Evaluation Metrics

To comprehensively evaluate the performance of the proposed multi-agent framework, we define the following metrics. These capture both the correctness and operational characteristics of the agents during validation tasks:

- **Per-Run Accuracy:** The percentage of individual runs (out of five per scenario) that resulted in the correct validation outcome.
- **Per-Scenario Accuracy:** The proportion of scenarios for which the correct outcome was obtained in at least 3 out of 5 runs.
- **Latency:** The average time (in seconds) taken by the system to complete a validation run, measured from task initiation to final judgment.
- **Number of Steps:** The average number of thought-action-observation cycles executed per run.
- **Number of Actions:** The average number of browser-based interactions (e.g., clicks, form submissions, navigations) taken by agents during a run.

5.3 Human Expert Validation

To ensure the reliability and interpretability of the agent-generated results, a human expert in enterprise accounting was engaged to perform a qualitative evaluation of the outputs. This expert was tasked with reviewing a subset of scenarios from both the Vanilla Agent and the proposed multi-agent framework. For each scenario, the expert assessed the reasoning process, the sequence of steps taken, the accuracy of the conclusions, and the presence of hallucinated content. The review focused on the following criteria:

- Reasoning Quality: Assesses whether the agent's logic reflects sound financial analysis and adheres to accounting principles. Ratings were assigned on a 1–5 scale:
  - 1: Severely flawed or illogical reasoning.
  - 2: Contains some logical structure but includes key errors.
  - 3: Generally logical reasoning; partial or incomplete use of available data.
  - 4: Sound reasoning with clear explanations and sufficient supporting data.
  - 5: Comprehensive and well-structured reasoning with nearly complete data retrieval and explicit, accurate calculations.
- Step Validity: Measures how accurately the agent's navigation and investigative steps mirror human analytical behavior. Ratings were also on a 1–5 scale:
  - 1: Steps are vague or missing.
  - 2: Only basic or generic actions are listed.
  - 3: Describes page sequence but lacks detail on investigative intent.
  - 4: Specifies data sought on each page, showing purposeful navigation.
  - 5: Clearly documents each visited page and justifies what was inspected or retrieved at each step.

The above metrics are included as part of the experimental analysis described subsequently.

5.4 Results and Analysis

To assess the effectiveness of the proposed multi-agent framework, we conduct a comparative evaluation against a baseline model referred to as the *Vanilla Agent*. As shown in Figure 5, the Vanilla Agent represents a single, general-purpose LLM-powered browser agent with unrestricted access to all simulated financial systems in the environment. Unlike the proposed approach, where each agent is specialized in a specific domain (e.g., GL, SL, Invoice Processing), the Vanilla Agent is tasked with performing the entire validation workflow independently, without modular delegation or domain-specific specialization.

This comparison aims to isolate the benefits of specialization and distributed reasoning in complex, multi-system financial environments. Both the Vanilla Agent and the multi-agent system were evaluated on an identical set of 20 validation scenarios, as described in Section 3, with each scenario executed five times to account for variability inherent in LLM outputs and web interactions.

As shown in Table 1, the results revealed the efficacy of the proposed multi-agent framework over the Vanilla Agent. In terms of per run accuracy, the multi-agent system achieved a score of 66%, compared to 52% for the Vanilla Agent, indicating a more consistent ability to arrive at correct conclusions in individual validation attempts. As depicted in Figure 3 the proposed agent demonstrated superior performance, accurately identifying 48 valid insights and 18 false alarms. In contrast, the Vanilla agent correctly identified only 39 and 13 instances, respectively. Similarly, the per-scenario accuracy, which measures whether the majority of runs per scenario yield the correct outcome, improved significantly from 40% under the Vanilla Agent to 65% with the proposed approach. However, some errors persisted in the multi-agent system, particularly in cases involving confusing entity names (e.g., "MEGA CORP" vs. "MEGA INDUSTRIES") or when the agent failed to fully explore relevant website sections. These cases highlight opportunities for further refinement in entity disambiguation and browsing strategies.

These improvements suggest that domain specialization and modular coordination between agents enable more reliable and accurate validation across complex, cross-system financial tasks. The increased per scenario accuracy further implies that the proposed

Table 1: Performance Comparison Between Vanilla and Proposed Multi-Agent Systems

Metric	Vanilla Agent	Proposed
Per-Run Accuracy	52%	66%
Per-Scenario Accuracy	40%	65%
Avg Reasoning Quality Score	3.05	3.60
Avg Step Validity Score	3.20	4.30
Latency (seconds)	60.80	324.91
Number of Steps	NA	3.43
Number of Actions	5.05	19.95

architecture is more robust to variability in LLM outputs and environmental noise, making it more suitable for high-stakes enterprise applications where consistency and correctness are critical.

The human expert evaluation further highlighted the benefits of the proposed multi-agent framework over the baseline. As shown in Table 1, the average Reasoning Quality Score for the multi-agent system was 3.60, representing an approximate 18% improvement over the Vanilla Agent's score of 3.05. This indicates that the multi-agent framework more consistently applied domain-relevant financial logic and provided clearer justifications for its conclusions. Additionally, the Step Validity Score showed an even greater improvement: the proposed approach achieved an average score of 4.30, compared to 3.20 for the Vanilla Agent (an approximate 34% increase). This demonstrates that the multi-agent system followed more deliberate and human-aligned investigative workflows, reflecting structured and purposeful navigation across financial systems. These qualitative gains are critical for enterprise applications, where traceability, transparency, and explainability are paramount.

In terms of operational behavior, the proposed multi-agent system demonstrated a more deliberate and comprehensive exploration of the environment compared to the Vanilla Agent. As shown in Table 1, the average number of actions taken by the proposed system was significantly higher (19.95) than that of the Vanilla Agent (5.05), indicating deeper interaction across systems and more thorough validation steps. While the number of intermediate reasoning steps was not applicable to the Vanilla Agent due to its monolithic structure, the multi-agent framework averaged 3.43 reasoning steps per scenario. This layered reasoning reflects the system's ability to coordinate across specialized agents, apply domain-specific logic, and build traceable evidence chains, behaviors essential for reliable root-cause analysis and auditability in enterprise settings.

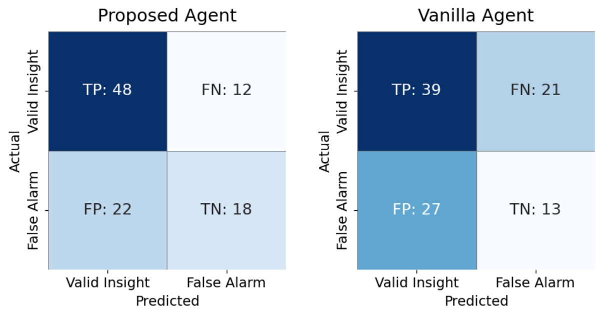


Figure 3: Performance of the proposed multi-agent system and the Vanilla agent across 100 validation runs (combining valid insights and false alarms) using the Per-Run Accuracy metric.

In terms of latency, a notable difference was observed between the two approaches. The proposed multi-agent system exhibited a significantly higher average latency of 324.91 seconds per scenario compared to just 60.80 seconds for the Vanilla Agent. This increase is expected due to the added overhead of coordination, planning, and inter-agent communication in the multi-agent setup. While the modular approach enhances accuracy and consistency, it introduces additional time costs stemming from sequential task execution and the overhead of orchestrating multiple specialized agents. This trade-off highlights the need to balance performance gains with operational efficiency, particularly in time-sensitive applications.

6 Conclusion

This work presents a novel multi-agent framework that leverages LLM-powered browser agents, each specialized in a distinct financial domain, to perform post-detection validation across simulated enterprise accounting systems. Unlike traditional monolithic approaches, the proposed system decomposes the validation task into domain-specific subtasks, enabling deeper reasoning, greater consistency, and enhanced auditability. Through rigorous scenario-based testing, the framework outperformed a general-purpose baseline agent in both accuracy and stability, while demonstrating more interpretable reasoning steps and a significantly lower hallucination rate. Importantly, validation by a human accounting expert confirmed that the multi-agent system produced more trustworthy and explainable outcomes, with better alignment to real-world accounting logic.

The proposed architecture highlights the value of specialization and modular coordination in complex, multi-system environments, especially where traceability and cross-system reconciliation are essential. By mimicking the workflows of expert financial analysts and auditors, the system bridges technical and semantic gaps across disparate systems without requiring centralized data pipelines. Future work will extend this approach to real-world enterprise datasets, integrate adaptive feedback loops, and explore broader applications in internal audit, compliance, and fraud detection.

References

[1] De Chezelles, Thibault Le Sellier, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F Xu, Siva Reddy, Quentin Cappart, et al. 2024. The BrowserGym ecosystem for web agent research. *arXiv preprint arXiv:2412.05467* (2024).

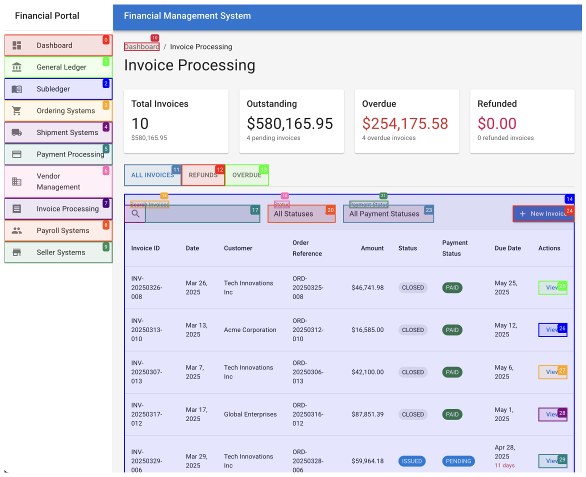
[2] Chanyeol Choi, Alejandro Lopez-Lira, Yongjae Lee, Jihoon Kwon, Minjae Kim, Juneha Hwang, Minsoo Ha, Chaewoon Kim, Jaeseon Ha, Suyeol Yun, et al. 2025. Structuring the Unstructured: A Multi-Agent System for Extracting and Querying Financial KPIs and Guidance. *arXiv preprint arXiv:2505.19197* (2025).

[3] Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, et al. 2024. AutoWebGLM: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5295–5306.

[4] Yuxuan Lu, Bingsheng Yao, Hansu Gu, Jing Huang, Jessie Wang, Yang Li, Jiri Gesi, Qi He, Toby Jia-Jun Li, and Dakuo Wang. 2025. UXAgent: A System for Simulating

Usability Testing of Web Design with LLM Agents. *arXiv preprint arXiv:2504.09407* (2025).
[5] Magnus Müller and Gregor Žunič. 2024. *Browser Use: Enable AI to control your browser.* https://github.com/browser-use/browser-use
[6] Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. 2024. WebCanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373* (2024).
[7] Frank F Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, et al. 2024. TheAgentCompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161* (2024).
[8] Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2024. AgentOccam: A simple yet strong baseline for llm-based web agents. *arXiv preprint arXiv:2410.13825* (2024).
[9] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023. WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854* (2023).

A Supplementary Figures



Invoice Processing

Total Invoices	Outstanding	Overdue	Refunded
10	$580,165.95	$254,175.58	$0.00
$580,165.95	4 pending invoices	4 overdue invoices	0 refunded invoices

Invoice ID	Date	Customer	Order Reference	Amount	Status	Payment Status	Due Date	Actions
INV-20250326-008	Mar 26, 2025	Tech Innovations Inc	ORD-20250325-008	$46,741.98	CLOSED	PAID	May 25, 2025	View 25
INV-20250313-010	Mar 13, 2025	Acme Corporation	ORD-20250312-010	$16,585.00	CLOSED	PAID	May 12, 2025	View 26
INV-20250307-013	Mar 7, 2025	Tech Innovations Inc	ORD-20250306-013	$42,100.00	CLOSED	PAID	May 6, 2025	View 27
INV-20250317-012	Mar 17, 2025	Global Enterprises	ORD-20250316-012	$87,851.39	CLOSED	PAID	May 1, 2025	View 28
INV-20250329-006	Mar 29, 2025	Tech Innovations Inc	ORD-20250328-006	$59,964.18	ISSUED	PENDING	Apr 28, 2025 11 days	View 29

Figure 4: Screenshot of the Invoicing Agent browsing the Invoice Processing page.

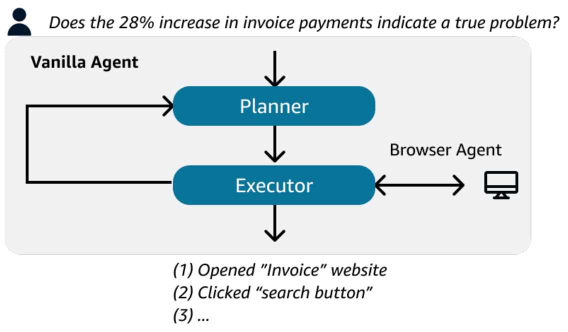

Does the 28% increase in invoice payments indicate a true problem?
Vanilla Agent
Planner
Executor
Browser Agent
(1) Opened "Invoice" website
(2) Clicked "search button"
(3) ...

Figure 5: The Vanilla Agent represents a single, general-purpose LLM-powered browser agent, implemented using BrowserUse [5] with unrestricted access to all simulated financial systems in the environment.

B Simulation Scenarios

The following are some of the scenarios used in the experimentation:

- The Accounts Payable system shows that 12 invoices from vendor GlobalTech Inc. were paid twice in April 2025, resulting in an overpayment of $157,850.
- The subledger flags an unusual pattern where the total daily revenue for February 28, 2025, is 45% lower than the expected value based on historical trends.
- The Ordering Systems report shows that 12 purchase orders totaling $143,500 were created for IT equipment from FastTech Supplies in March 2025, but the General Ledger shows no corresponding fixed assets additions or expense entries for these purchases.
- The Payroll Systems report for April 2025 shows 15 employees received duplicate bonus payments totaling $87,500, with the second payment occurring exactly one week after the first legitimate payment.
- The Payroll Systems report shows an unusual 18% increase in total payroll expenses for the North American region in May 2025 compared to April 2025, despite no reported increase in headcount.

C Prompts

C.1 Planner

```
You are an accounting assistant tasked with helping the
accountant determine the validity of system insights.
This research requires analysis of multiple systems to
create a holistic picture of the system. For the given
insight determine a simple step by step plan. The plan
should have ordered individual tasks that when executed
correctly will let you determine if an insight is valid
or not. Do not add any unneccessary steps. All
transactions in the system should flow through the
Subledger and General Ledger so you should validate the
data in those systems in all scenarios. You have access
to assistants that are system experts who can complete
research on each of the individual accounting systems
as follows:
{agent_descriptions}

Output only the plan and nothing else.
{format_instructions}
```

C.2 Replanner

```
For the given objective, come up with a simple step by
step plan. This plan should involve individual tasks,
that if executed correctly will yield the correct answer.
Do not add any superfluous steps. The result of the final
step should be the final answer. Make sure that each step
has all the information needed - do not skip steps.

Your objective was this:
```

```
{input}

Your original plan was this:
{plan}

You have currently done the follow steps:
{past_steps}

Update your plan accordingly. If no more steps are needed
and you can return to the user, then respond with that.
Otherwise, fill out the plan. Only add steps to the plan
that still NEED to be done. Do not return previously done
steps as part of the plan.
```

C.3 Supervisor Response

```
Using the data collected for you, provide a final analysis
of if the insight is a true or false positive. A true
positive means the insight was correct or a similar issue
is found. For example if the numbers do not match exactly
but the issue still exists, report true positive but
indicate manual follow up is required.
A false positive means the issue mentioned does not exist
at all.
Make the conclusion "Needs Manual Review" if you cannot
validate the insight due to missing data or are unsure
about the final judgement.

Here is the initial result from your subordinates:
{response}

As a reminder the initial insight was:
{insight}

The steps executed to find data were:
{past_steps}

Format your ouput like the following:
<example>
Conclusion: True Positive/False Positive/Needs Manual Review

Reasoning and evidence:
    <outline your reasoning and provide evidence>
    1. ...
    2. ...

Steps:
    <explain the steps you took to find the result>
    1. ...
    2. ...
</example>
```