# Replication as Learning: Scalable Knowledge Distillation for Multimodal Enterprise Agents

**Xiaoli Zhang** [1] **Elham Alipour** [1] **Tom Boyang Jin** [1] **Alex Moschos** [1] **Eugene Kim** [1] **Miriam Teng** [1]

## Abstract

Enterprise environments differ fundamentally from the clean settings assumed in LLM research: knowledge is distributed across heterogeneous sources, often incomplete or inconsistent, and key procedural logic is implicitly encoded in artifacts rather than explicitly documented. In such settings, retrieval-based approaches are insufficient, as no single source contains the full workflow.

We propose a replication-driven knowledge distillation framework for scalable learning in multimodal agents. The agent learns by reverse-engineering validated artifacts (e.g., Excel workbooks), reconstructing the underlying data pipeline, and distilling the inferred logic into structured knowledge (claims, procedures, and domain patterns). This enables synthesis and validation across noisy sources and supports reuse in future tasks.

We evaluate on 120 simulated enterprise environments with multimodal inputs (SQL, spreadsheets, documentation, messaging app, emails, images, PDFs, CSV) and controlled noise. Our method consistently outperforms retrieval-based baselines on both task execution and conceptual understanding, and remains robust under environmental drift.

## 1. Introduction

Enterprise environments present a fundamentally different challenge from the clean, well-structured settings commonly assumed in LLM agent (Yao et al., 2023) research. Knowledge is distributed across heterogeneous sources (e.g., spreadsheets, databases, documentation, and human communication), often incomplete or inconsistent, and critical procedural logic is frequently implicit rather than explicitly documented (Dong et al., 2024) (Hernes et al., 2020). Importantly, no single source contains the full end-to-end workflow. Even validated "golden" artifacts, such as CFO-signed-off Excel workbooks, encode only partial logic, while key steps (e.g., data extraction or filtering rules) must be inferred from other noisy sources or reconstructed through reasoning. As a result, agents must go beyond retrieval (Lewis et al., 2020) (Gao et al., 2023) and reconstruct workflows via iterative validation.

We propose a replication-driven knowledge distillation framework for scalable learning in multimodal agents. Instead of relying on curated documentation, the agent learns by reverse-engineering validated artifacts, reconstructing the underlying data pipeline, and resolving inconsistencies across sources. The inferred logic is distilled into structured representations, *claims*, *procedures* and *domain knowledge*, which are directly reusable and human-interpretable. This design is inherently scalable in enterprise settings: obtaining validated artifacts (e.g., signed-off files) is low-cost, while producing clean documentation is often impractical. Moreover, the distilled outputs (e.g., structured markdown files and script) can be easily inspected and refined by users, enabling efficient human-in-the-loop improvement.

We study this problem in the context of account reconciliation, a common enterprise task requiring analysts to verify consistency across financial systems while explaining expected deviations (e.g., timing differences or adjustments). The task involves data extraction, spreadsheet construction, and judgment under uncertainty, making it a representative setting for multimodal procedural learning (Appendix A provides more background on account reconciliation task).

To evaluate our approach, we construct a controlled simulation environment that mirrors real enterprise workflows, where agents interact with multimodal tools including SQL, spreadsheets, messaging app, email, and file systems containing both validated artifacts and noisy knowledge sources. Knowledge is intentionally fragmented and corrupted, requiring agents to integrate and infer across modalities rather than rely on any single source.

We evaluate a coding agent on two tasks: (1) *artifact construction*, which tests end-to-end execution by requiring

[1]Amazon. Correspondence to: Xiaoli Zhang <zhasabri@amazon.com>.

*Proceedings of the $43^{rd}$ International Conference on Machine Learning*, Seoul, South Korea. PMLR 306, 2026. 

the agent to generate a reconciliation workbook for a new period, and (2) *Q&A*, which probes deeper understanding of the underlying logic and data. Together, these tasks assess both procedural execution and conceptual reasoning, corresponding to senior-level analyst capabilities. We compare against a retrieval-style baseline implemented as direct access to the original file system, rather than a conventional RAG pipeline. In our setting, generic RAG performance would be dominated by challenges in parsing and indexing complex spreadsheet elements (e.g., formulas, comments, embedded images), whereas our focus is on comparing retrieval-based access versus knowledge distillation.

In summary, this paper makes three contributions. First, we develop a framework for simulating noisy enterprise environments, where knowledge is incomplete, incorrect, and distributed across artifacts and communication channels. Second, we propose a replication-driven distillation method that learns procedural enterprise knowledge by reconstructing reconciliation workbooks, rather than relying on retrieval from raw documentation alone. Third, we show that distilled knowledge can serve as a guardrail for future execution, enabling robust adaptation under environmental changes such as schema drift, aggregation changes, and ingestion failures.

## 2. Related Work

**Spreadsheet and Excel agents.** Recent work studies LLMs as agents for spreadsheet manipulation and understanding. *SheetCopilot* formulates spreadsheet control as atomic actions (Li et al., 2023). *SpreadsheetAgent* proposes multi-agent reasoning over spreadsheet content and layout (Ren et al., 2026). *SheetMind* uses manager, action, and reflection agents for spreadsheet task execution (Zhu et al., 2025). These works focus primarily on inference-time spreadsheet understanding or control, whereas we study learning enterprise procedures through workbook replication and distilling them into reusable knowledge under noisy artifacts.

**Coding agents and long-horizon task execution.** Our setting is also related to coding agents, where artifact construction requires iterative validation under complex constraints. *SWE-EVO* emphasizes long-horizon software engineering over isolated bug fixes (Thai et al., 2025), while *Confucius Code Agent* studies scaffolding for agents operating over large codebases (Wong et al., 2025). These works highlight long-horizon artifact generation, but focus on software repositories rather than spreadsheet-based business workflows with incomplete or outdated documentation.

**Agents under noisy or imperfect environments.** *AgentNoiseBench* studies robustness of tool-using agents under user and tool noise (Wang et al., 2026). While closely related in motivation, it focuses on robustness degradation, whereas we study how agents can acquire stable procedural knowledge despite incomplete, incorrect, and contradictory enterprise information.

**Enterprise workflow agents and orchestration.** *Orchestrating Agents and Data for Enterprise* studies structured coordination across agents, tools, and proprietary data (Kandogan et al., 2025), while *SOAN* explores self-organizing multi-agent workflows (Xiong et al., 2025). These works focus on enterprise orchestration, whereas we focus on learning, distilling, and reusing procedural reconciliation knowledge in noisy enterprise environments.

## 3. Simulation Environment

### 3.1. Overview

Figure 1 illustrates how the simulated enterprise environment is constructed. Our goal is to approximate the working conditions of a human analyst who must recover a real business workflow from incomplete and unreliable evidence.

At a high level, we begin with a hidden end-to-end reconciliation process that transforms raw transaction data (subledger (SL) data from multiple tables) into final summarized outputs (general-ledger (GL) data in a single table)[1]. This process can be viewed as a latent computation pipeline: it specifies how different data sources relate to one another, how values are filtered or aggregated, and how recurring patterns such as weekend effects, month-end effects, or beginning-of-month effects arise.

Rather than exposing this process directly, we construct the observable environment through two complementary components. First, we retain a validated *golden* Excel reconciliation workbook[2] that serves as a correct downstream reference. The workbook always contains correct final outputs and may expose some local calculation logic, but it is not a complete specification of the full process. Upstream steps may be absent, some logic may be obscured through hard-coded values, and some information may appear only in non-tabular forms such as screenshots. Second, we distribute partial knowledge of the hidden process across multiple external sources, including documentation files, SQL history, and simulated human contacts through messaging app and email. These sources may be incomplete, outdated, or incorrect.

Agents are given access to (i) the golden Excel workbook, (ii) multiple versions of documentation, and (iii) enterprise-style tools including a SQL interface, communication chan-

[1] See Figure A in Appendix A for additional background on the reconciliation process and the roles of GL and SL systems.

[2] See Appendix B for more details on why a golden file is required.

nels (Message/Email), and an Excel compute engine.[3]

A key property of the environment is that knowledge is inherently fragmented. No single source provides a complete and reliable description of the workflow. The workbook provides correct outputs but only partial logic; documentation exists in multiple inconsistent versions; SQL history contains both useful and irrelevant queries without reliable indicators of correctness; and human sources provide incomplete, uncertain, and role-limited information. As a result, agents must reconstruct the workflow through cross-source reasoning and iterative validation rather than by reading a single authoritative specification. For data patterns (e.g. seasonality), we do not provide it in any location, agent is expected to proactively infer data patterns and form domain knowledge by itself.

Agents construct Excel workbooks programmatically via a coding interface (e.g., file operations and script execution). These capabilities are provided by the agent framework and are not themselves part of the simulation environment, but they are required for task execution.

### 3.2. Files and Tools

The environment provides two types of resources: *artifacts* and *tools*.

Artifacts include multi-sheet Excel workbooks and multiple versions of supporting documentation. The Excel workbooks contain accounting data drawn from multiple systems, manual adjustments, and reconciliation summaries. Documentation files contain partial descriptions of procedures, data dependencies, and business rules, but may vary in quality and correctness.

Tools include a SQL interface (`query`, `describe_table`, `search_history`), communication channels for interacting with simulated colleagues, and an Excel compute engine for validating generated outputs. Together, these tools allow the agent to inspect data, test hypotheses, request missing information, and verify whether a generated workbook reproduces the expected outputs.

### 3.3. Coding Agent

We use a coding agent built on Claude Opus 4.5 to construct and validate Excel artifacts. The agent is equipped with standard programming capabilities, including file operations and code execution, and may delegate subtasks to a sub-agent with identical capabilities, except that recursive delegation is not allowed.

We emphasize that the focus of this work is not the specific design of the coding agent itself, but the learning paradigm built on top of it. The coding interface serves as a generic execution layer and could be replaced by alternative implementations without changing the core experimental setting.

### 3.4. Data Simulation

We simulate transaction data from two classes of accounting systems: General Ledger (GL) and Sub-ledger (SL) systems. In simple terms, SL systems record detailed operational transactions, while the GL stores the summarized balances that appear in financial reporting. The reconciliation task is to determine how these detailed and summarized views should align[4] .

Our simulation is based on real-world datasets with sensitive information masked and values scaled. The simulated data preserves several properties that make enterprise workflows challenging in practice, including heterogeneous schemas across systems, recurring seasonal patterns such as month-end effects, and expected deviations such as timing differences and manual adjustments. These deviations are important because they ensure that reconciliation requires judgment and causal reasoning rather than a simple equality check between multiple tables.

In addition to simulating raw data, we also derive recurring data-pattern configurations directly from the underlying process. Examples include weekend effects, month-end spikes, and beginning-of-month reversals. These patterns provide important indirect evidence that agents may use when explicit procedural knowledge is incomplete.

### 3.5. Noise Simulation

To model the uncertainty of real enterprise environments, we introduce controlled noise into the knowledge sources.

We begin with the ground-truth reconciliation process and decompose it into discrete *knowledge units*, where each unit represents a small piece of information such as a field mapping, a filtering condition, a query pattern, a transformation step, or an explanation for a recurring adjustment. Each knowledge unit is then assigned to one or more observable sources, such as documentation, SQL history, or simulated colleagues [5].

Given predefined missing and error rates, each (knowledge unit, source) pair is assigned one of three states: *correct*, *incorrect*, or *missing*. A correct unit is preserved as-is.

[3]Libraries such as `openpyxl` do not evaluate Excel formulas, which limits an agent's ability to inspect the values of the workbook it generates. We observed substantial improvements in accuracy for both baseline and proposed methods after adding an Excel compute engine that evaluates generated spreadsheets and exports computed outputs (e.g., CSV, PDF, PNG).

[4]See Figure A in Appendix A for additional background on the reconciliation process and the roles of GL and SL systems.

[5]see Appendix C for examples of noise configurations

An incorrect unit is replaced by a plausible but wrong alternative, such as substituting `transaction_date` for `posting_date`. A missing unit is omitted entirely from that source.

The final environment is constructed by aggregating these source-specific corrupted knowledge units across all sources. Importantly, missing information does not imply that a task is unsolvable. It only means that the corresponding knowledge is not explicitly available in that particular source and must instead be inferred from other evidence, such as SQL history, workbook structure, or consistency with observed outputs.

The golden workbook itself is never corrupted. It provides a stable downstream reference that keeps tasks solvable while still requiring the agent to recover missing logic through hypothesis generation, cross-source comparison, and iterative validation.

# 4. Methodology

## 4.1. Overview

We propose a replication-driven knowledge distillation framework for learning procedural knowledge in noisy enterprise environments. The framework consists of two stages. In the *learning stage*, the agent reverse-engineers a validated artifact (the golden workbook) and distills the underlying logic into structured knowledge. In the *execution stage*, the agent is given only the distilled knowledge and must perform the same task for a new time period. This separation allows us to evaluate whether the agent has learned generalizable procedures rather than memorizing specific instances.

## 4.2. Replication as Learning

In our setting, no single source contains the full procedure, and even the golden artifact encodes only partial logic. We therefore formulate learning as a *replication problem*, where the agent reconstructs the process that generates the artifact.

The agent produces an executable program that extracts data, applies transformations, and generates an Excel workbook matching the golden output. Correctness is verified by comparing computed outputs against the original artifact. This induces an iterative loop of proposing candidate logic (e.g., queries, mappings, formulas), executing it, and refining it based on discrepancies. Through this process, the agent resolves inconsistencies and infers missing steps, effectively learning via hypothesis testing and validation rather than retrieval (Lewis et al., 2020) (Appendix D provides the prompts).

## 4.3. Knowledge Distillation

The inferred logic is distilled into three components: *claims*, *procedures*, and *domain knowledge*. Claims represent atomic facts with uncertainty and evidence; procedures encode the end-to-end workflow along with executable scripts; and domain knowledge captures higher-level patterns such as expected ranges and typical deviations. This structure separates factual grounding, execution logic, and judgment, producing a representation that is both human-interpretable and directly reusable.

## 4.4. Execution and Generalization

In the execution stage, the agent uses only the distilled knowledge to construct a reconciliation workbook for a new period and answer questions about the process. It does not have access to the original artifact or external sources, ensuring that performance reflects internalized knowledge rather than retrieval.

## 4.5. Knowledge as Guardrail

Distilled knowledge also serves as a mechanism for verification and adaptation. When the environment changes (e.g., schema updates or data issues), the agent converts its knowledge into validation rules to detect inconsistencies and adjust its workflow. This mirrors how experienced analysts rely on internalized expectations to identify anomalies. As a result, knowledge functions not only as memory, but as a constraint system guiding reasoning under uncertainty. We demonstrate this concept through robustness tests.

# 5. Evaluation

## 5.1. Evaluation Setup

We evaluate whether the proposed framework enables agents to learn actionable, in-depth, and robust procedural knowledge. Each experiment batch consists of 120 simulated environments, spanning multiple business domains. Each environment includes a golden Excel workbook constructed using one of four common enterprise formats (e.g., pivot-based summaries, screenshot-based inputs, hardcoded values, and comment-based metadata). These variants introduce structural diversity and reflect real-world spreadsheet practices. See Excel formatting details in Table 5 and screenshots of workbooks in Figure 3.

Following the two-stage protocol described in Section 4, evaluation only focues on the output from execution stage. We evaluate under three noise settings, clean, medium, and high, with 0%, 25%, and 50% missing and incorrect knowledge respectively). We use the raw file system as the primary retrieval-style baseline, rather than a conventional RAG stack, because our goal is to isolate the effect of knowledge

representation rather than confound it with variability in document parsing, chunking, and embedding quality. In our setting, the environment includes heterogeneous artifacts such as large Excel sheets with formulas, cell comments, embedded images, and screenshot-only data sources that are not reliably handled by most off-the-shelf RAG systems without substantial task-specific engineering. Benchmarking against a generic RAG pipeline would therefore primarily measure limitations in artifact ingestion, rather than the distinction we study between retrieval-based access and replication-driven knowledge distillation. Since improving multimodal parsing or retrieval infrastructure is not the focus of this work, we use direct access to the original file system as a stronger and fairer retrieval-style baseline, giving the baseline access to the raw artifacts without introducing avoidable losses from imperfect preprocessing.

### 5.2. Evaluation Tasks

We use two complementary tasks to evaluate both execution capability and conceptual understanding.

The **Excel task** requires the agent to construct a reconciliation workbook for a new period. This reflects the core responsibility of a financial analyst and evaluates whether the agent can correctly reconstruct and execute the end-to-end workflow.

The **Q&A task** probes deeper understanding of the process and data (See sample questions in Appendix F), including identifying correct queries, explaining discrepancies, and reasoning about expected deviations. Questions are designed to require deriving the correct answer rather than validating a given source, forcing the agent to reason across noisy inputs. This task reflects the level of understanding expected from a senior analyst.

### 5.3. Ablation Study

We perform prompt-level ablations to isolate key components of the framework under high-noise settings.

**No Replication** removes execution-based learning, requiring the agent to summarize knowledge without reconstructing the workflow, testing the necessity of learning through execution.

**No Claims** remove structured `claims.md` and just use knowledge and procedure, evaluating the role of uncertainty-aware memory.

**Single-Agent Distillation** replaces the multi-agent distillation process with a single agent, testing the benefit of role specialization.

### 5.4. Robustness Evaluation

We evaluate robustness under three environment changes commonly seen in large enterprise in the high-noise setting.

**Schema Drift:** column renaming causes queries to return empty results for one field for the new period, requiring agent to identify the new field on its own.

**Aggregation Change:** previously merged SL sources appear separately, breaking GL–SL mappings.

**Ingestion Failure:** SL transactions from a specific category are missing for 5 days, creating subtle discrepancies with GL.

We report a **drift detection rate**, indicating whether the agent identifies the underlying issue in its summary.

### 5.5. Metrics

#### 5.5.1. Excel Evaluation

Evaluating Excel outputs is challenging due to implementation flexibility.[6] We therefore use rule-based metrics aligned with human judgment.

**Accuracy** measures numerical correctness as the fraction of ground-truth cells whose evaluated values match the agent output:

$$\text{Acc} = \frac{\sum_i \mathbf{1}(v_i = v_i^*)}{N},$$

where $v_i$ and $v_i^*$ are the evaluated values of the agent and ground-truth workbooks, respectively. We first use the Excel compute engine to materialize both workbooks into CSV files before comparison. Cells in raw data sheets are excluded, since agents may import extra columns without affecting the reconciliation logic.

**Formula usage ratio** measures whether the agent uses formulas rather than hard-coded values:

$$\text{Formula} = \frac{\#\{\text{formula cells in agent workbook}\}}{\#\{\text{formula cells in ground truth workbook}\}}.$$

We do not require exact formula match, since equivalent formulas may take many forms. Instead, this metric checks whether the agent preserves formula-based computation, which is critical for auditability in enterprise spreadsheets. To prevent trivial cheating, we additionally verify that formulas are semantically meaningful.

**Layout similarity** measures whether the agent preserves the expected spreadsheet structure. For each sheet, we compare the value range occupied by the agent output with the

[6] Agents are instructed to follow the exact template of the golden workbook. This reflects real-world practice, where repeated enterprise tasks typically rely on standardized Excel templates, and also makes rule-based evaluation feasible.

ground-truth range (e.g., whether the main table appears in the same region such as A1:F9). This captures output stability, which is important because downstream workbooks often depend on fixed sheet layouts.

**Formatting score** measures presentation quality using four equally weighted binary checks: number formatting, column width, cell coloring, and text styling. This metric captures whether the workbook is formatted to be human-readable and visually aligned with the template. Unlike layout, which measures structural stability, formatting focuses on presentation quality.

We combine these metrics into an overall **Excel score**:

Score = 0.7 Acc + 0.2 Formula + 0.05 Layout + 0.05 Format.

This weighting prioritizes numerical correctness while rewarding formula-based computation and consistency with the expected template.

#### 5.5.2. Q&A Evaluation

For the Q&A task, we measure accuracy using an LLM-based judge, with manual verification applied to a subset of cases.

#### 5.5.3. Evaluation Design and Validation

We developed the evaluation framework iteratively through multiple rounds of refinement and human review. Metrics were adjusted to align with human judgment of spreadsheet quality, particularly for handling equivalent formulas and flexible implementations. We manually reviewed over 10% of results and agent operation logs to validate correctness and identify edge cases.

We also track auxiliary signals, including LLM-based scoring and keyword-based Q&A accuracy. We explored using an LLM-agent-based score as a reference metric by providing the agent with both generated and ground-truth workbooks, along with computed outputs (CSV) and a scoring rubric. However, we observed high variability in scoring across cases. For example, the same underlying error (e.g., using `posting_date` instead of `transaction_date`) could be penalized inconsistently depending on whether it was treated as a single logical mistake or multiple cell-level errors. Due to this instability and the difficulty of validating evaluation reasoning at scale, we do not use this metric for final evaluation. Nevertheless, the score and accompanying reasoning remain useful for identifying edge cases and guiding manual review, to flag inconsistencies and trigger further inspection.

Safeguards against cheating like formula keyword check are included by verifying that formulas are semantically meaningful (e.g., preventing trivial equality checks or circular references).

### 5.6. Quantitative Results

Table 1 and Figure 4 shows that distillation consistently outperforms the retrieval-based baseline across all noise levels. Under high noise, Excel score improves from 75 to 89, driven by higher formula usage and improved accuracy, while Q&A accuracy increases from 82 to 88. Performance remains stable as noise increases, indicating robustness to missing and incorrect knowledge.

Formatting (e.g bold text, cell width) is worse in the distilled setting, as the agent does not have access to original files for direct copying. Despite this, distilled knowledge achieves both higher correctness and stronger adherence to best practices (e.g., formula usage).

In terms of efficiency, distilled knowledge uses 80–90% of baseline tokens for Excel tasks and 40–50% for Q&A. We observe that baseline agents often fail to use formulas despite explicit instructions, likely due to attention dilution after exploring noisy sources, leading to fallback behaviors such as precomputing values outside Excel.

Ablation results (Table 7) confirm that replication is critical (Excel score drops from 89.1 to 83.7 without replication), while removing claims or using a single-agent distillation slightly degrades performance. Under drift scenarios (Ttable 8), the agent maintains strong Excel performance (88–91) and achieves high detection rates (up to 98%), demonstrating that distilled knowledge functions as an effective guardrail.

### 5.7. Qualitative Observations

#### 5.7.1. Common Failure Modes

**Incorrect field selection under subtle deviations.** A recurring failure mode arises when the agent relies on incorrect SQL queries provided by documentation or colleagues, especially when both sources contain incorrect info. In particular, agents sometimes use `posting_date` instead of `transaction_date` when performing daily reconciliation. Under high-noise conditions, cross-referencing multiple sources is often insufficient to resolve this ambiguity, as both fields are plausible and frequently appear in documentation. Due to business characteristics, these fields produce numerically similar outputs, with deviations that are small (e.g., $<5\%$) and therefore difficult to detect. In such cases, correct resolution requires deeper reasoning, such as tracing how dates are used in the formulas of the golden workbook. This highlights a limitation of surface-level validation and the need for artifact-grounded reasoning.

**Failure to adapt formulas to reconstructed data pipelines.** Another observed failure mode occurs when the agent correctly reconstructs data extraction logic but fails to adapt downstream Excel formulas accordingly. For example, if

*Table 1.* Main results across all experiment settings. Excel Score is a weighted combination of sub-metrics (see Section 5).

| Noise Level | Environment | Excel Score | Q&A Acc. | Accuracy | Formula Usage | Layout | Format |
|---|---|---|---|---|---|---|---|
| Clean | Distilled Knowledge | 89.4 | 87.5 | 92.7 | 96.2 | 88.5 | 18.0 |
| | Original File System | 78.7 | 83.7 | 91.8 | 37.2 | 86.0 | 53.1 |
| Medium | Distilled Knowledge | 89.3 | 87.4 | 92.3 | 95.4 | 87.2 | 25.7 |
| | Original File System | 75.4 | 81.4 | 89.3 | 30.8 | 86.7 | 46.5 |
| High | Distilled Knowledge | 89.1 | 88.0 | 91.9 | 96.0 | 88.3 | 22.7 |
| | Original File System | 75.4 | 82.2 | 87.8 | 35.2 | 86.6 | 51.6 |

the generated dataset places the amount field in a different column position, the agent may still copy formulas from the golden workbook without updating cell references (e.g., using column B instead of column C). This error typically results in large discrepancies in reconciliation outputs, which often triggers subsequent debugging by the agent. As a result, this failure mode is less persistent, but it reveals a tendency toward surface-level copying rather than full reverse engineering.

**Failure to validate all patterns.** For ingestion failure scenario in robustness experiment, the detection rate (71.7%) is relatively low. Log analysis shows the agent did document the knowledge in learning stage and did review the data pattern as instructed in execution stage but fail to notice the issue of zero transactions for 5 days in the middle of the month because it was focusing too much on confirming the month beginning and month end patterns. We failed to find a better prompt or process (e.g. ask agent to compile valdiation.py and/or validation.md, single agent vs more sub-agent) that can fix the issue, but we can achieve a detection rate of 93.3% if we use Opus 4.6 instead.

## 6. Conclusion

We study the problem of learning procedural knowledge in enterprise environments, where information is fragmented, noisy, and often implicitly encoded in artifacts rather than explicitly documented. In such settings, retrieval-based approaches are insufficient, as no single source contains the complete and correct workflow.

This work makes three main contributions. First, we develop a framework for systematically simulating noisy enterprise environments, in which knowledge is incomplete, incorrect, and distributed across heterogeneous artifacts and communication channels. This simulation setting captures key challenges of real-world enterprise workflows, including fragmented documentation, unreliable human input, and partially observable processes.

Second, we propose a replication-driven knowledge distillation approach, in which agents learn procedural knowledge by reconstructing validated artifacts (e.g., reconciliation workbooks), rather than relying solely on retrieval from raw documentation. This paradigm enables agents to move beyond passive lookup, instead learning through hypothesis generation, validation, and synthesis across sources. The resulting knowledge, organized as claims, procedures, domain patterns, and executable scripts -supports both execution and reasoning under uncertainty.

Third, we show that the distilled knowledge functions as a guardrail for future execution. By proactively learning domain patterns (e.g., seasonality and expected value ranges), the agent can detect anomalies and adapt to environmental changes such as schema drift, aggregation changes, and data ingestion failures.

Through experiments in a controlled simulation environment, we demonstrate that the proposed approach substantially outperforms retrieval-based baselines across both execution (Excel tasks) and conceptual understanding (Q&A tasks). Importantly, performance remains strong even under high levels of noise, highlighting the robustness of the learned knowledge representations.

More broadly, our results suggest a shift from static, retrieval-based systems toward learning agents that construct, refine, and operationalize knowledge over time. While this work focuses on financial reconciliation as a representative task, the underlying framework is applicable to a wide range of enterprise workflows where knowledge is distributed, implicit, and evolving.

Future work includes extending this framework to continuous learning settings, improving knowledge updating under repeated interactions, and exploring tighter integration between structured knowledge representations and model training.

## References

Dong, S., Sahri, S., and Palpanas, T. Data quality awareness: A journey from traditional data management to data science systems. *arXiv preprint arXiv:2411.03007*, 2024.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y.,

Dai, Y., Sun, J., Guo, Q., Wang, M., and Wang, H. Retrieval-augmented generation for large language models: A survey. *ArXiv*, abs/2312.10997, 2023. URL https://api.semanticscholar.org/CorpusID:266359151.

Hemati, H., Schreyer, M., and Borth, D. Continual learning for unsupervised anomaly detection in continuous auditing of financial accounting data. *ArXiv*, abs/2112.13215, 2021. URL https://api.semanticscholar.org/CorpusID:245502006.

Hernes, M., Bytniewski, A., Matenczuk, K., Rot, A., Dziuba, S. T., Fojcik, M., Bernat, K., and Kozina, A. Data quality management in erp systems—accounting case. In *International Conference on Computational Collective Intelligence*, pp. 377–389, Cham, 2020. Springer.

Kandogan, E., Bhutani, N., Zhang, D., Chen, R. L., Gurajada, S., and Hruschka, E. Orchestrating agents and data for enterprise: A blueprint architecture for compound ai. *arXiv preprint arXiv:2504.08148*, 2025.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Li, H., Su, J., Chen, Y., Li, Q., and Zhang, Z. Sheetcopilot: Bringing software productivity to the next level through language-based spreadsheet manipulation. *arXiv preprint arXiv:2305.19308*, 2023.

Ren, H., Mingjie, Z., Lu, Z., Wang, K., and Yang, Y. Towards robust real-world spreadsheet understanding. *arXiv preprint arXiv:2604.12282*, 2026.

Schreyer, M., Sattarov, T., Schulze, C., Reimer, B., and Borth, D. Detection of accounting anomalies in the latent space using adversarial autoencoder neural networks. *ArXiv*, abs/1908.00734, 2019. URL https://api.semanticscholar.org/CorpusID:199405343.

Thai, M. V. T., Le, T., Manh, D. N., Nhat, H. P., and Bui, N. D. Q. Swe-evo: Benchmarking coding agents in long-horizon software evolution. *arXiv preprint arXiv:2512.18470*, 2025.

Wang, R., Chen, Y., Wang, Y., Wu, C., Fang, J., Cai, X., Gu, Q., Su, H., Zhang, A., Wang, X., Cai, X., and Chua, T.-S. Agentnoisebench: Benchmarking robustness of tool-using llm agents under noisy condition. *arXiv preprint arXiv:2602.11348*, 2026.

Wong, S., Qi, Z., Wang, Z., Hu, N., Lin, S., Ge, J., Gao, E., Chen, W., Du, Y., Yu, M., and Zhang, Y. Confucius code agent: Scalable agent scaffolding for real-world codebases. *arXiv preprint arXiv:2512.10398*, 2025.

Xiong, Y., Wang, J., Zhu, Y., and Zhao, Y. Self-organizing agent network for llm-based workflow optimization. *arXiv preprint arXiv:2508.13732*, 2025.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *ICLR*. OpenReview.net, 2023. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2023.html#YaoZYDSN023.

Zhu, R., Cheng, X., Liu, K., Zhu, B., Jin, D., Parihar, N., Xu, Z., and Gao, O. Sheetmind: An end-to-end llm-powered multi-agent framework for spreadsheet automation. *arXiv preprint arXiv:2506.12339*, 2025.

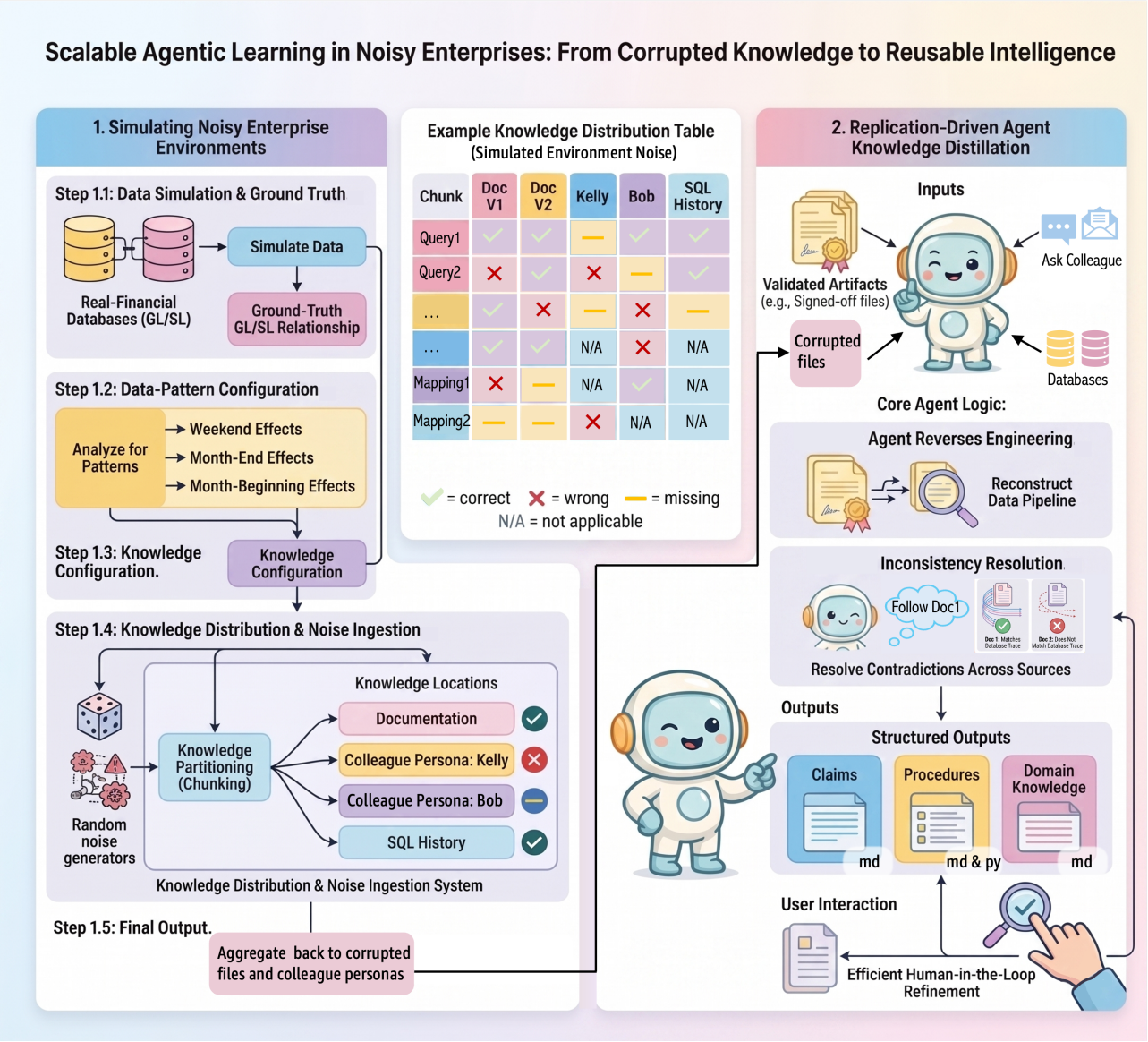



*Figure 1.* Overview of the simulated enterprise environment. We begin with a hidden reconciliation process that transforms raw transactional data (multiple sub-ledger data tables) into final outputs (a single general-ledger table), which defines recurring patterns and dependencies across data sources. We additionally introduce realistic deviations (e.g., manual adjustments and timing differences), so that reconciliation requires pattern-based reasoning and judgment rather than direct equality checks. To construct the observable environment, we decompose the hidden process into small knowledge units and distribute them across multiple sources, such as documentation, SQL history, and simulated colleagues. Each source may contain correct, incorrect, or missing information depending on the environment's noise level. In parallel, the Excel reconciliation workbook is retained as a correct but incomplete artifact: it contains valid final outputs and may reveal some local logic, but it does not fully specify the upstream process. The agent must therefore recover the missing workflow by combining fragmented evidence across sources and validating hypotheses against the workbook. Beyond reconstructing procedural logic, the agent must also proactively learn recurring data patterns (e.g., seasonality and value ranges), forming domain knowledge that serves as guardrails for detecting anomalies and adapting to future environment drift.

## A. Background on Account Reconciliation

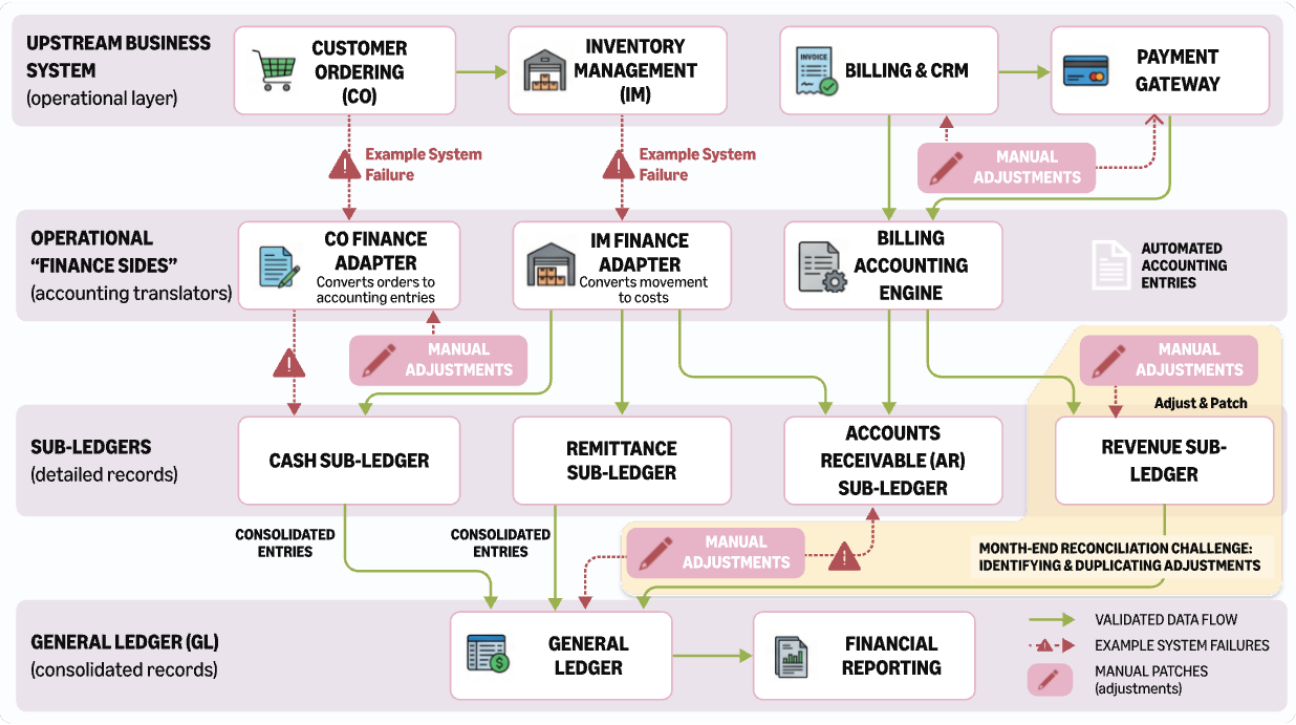



*Figure 2.* Illustration of large enterprise finance system. Finance system typically has multiple layers (Hemati et al., 2021), (Schreyer et al., 2019). The decentralized operational layer captures raw transaction events (e.g. customer ordering, inventory movements). Translation layer map those events to accounting entries (credits and debits) and ingest to sub-ledgers, which contain high-granularity records. Finally, sub-ledger data is standardized, merged and aggregated to general ledger. The GL is the "ultimate source of truth" used to generate external financial statements. A data flow failure can occur between any two layers. For example, a Billing system failure (Layer 1) might mean transactional data is never sent to the Translation Layer (Layer 2). Due to the tight timeline to release financial results, automated pipeline typically cannot be re-run for that specific day or event. Instead, an accountant must manually calculate and inject a corrective patch (the Blue Manual Adjustment boxes) directly into the affected layer (the Sub-Ledger or the GL) to resolve the state mismatch. This necessity for human-in-the-loop state correction, potentially requiring identify silent failures within the 5-7 day month-end close cycle, apply duplicated adjustments across multiple connected sub-systems, represents the primary bottleneck and complexity during enterprise accounting reconciliation work.

To make our setting concrete, we introduce the domain of account reconciliation, a standard financial process used across virtually all organizations.

At a high level, large companies maintain multiple financial systems. The General Ledger (GL) is the authoritative system used for financial reporting, while Sub-Ledgers (SL) represent operational systems (e.g., bank transactions, payment processors, or customer remittance systems). Because these systems operate independently, their balances often do not match exactly at any given point in time.

The goal of reconciliation is to verify that these systems are consistent after accounting for known differences. Importantly, reconciliation does not require exact equality between systems. Instead, differences must be explained and justified. For example:

- **Timing differences**: Transactions recorded in one system may appear in another system with a delay (e.g., bank processing delays).
- **Manual adjustments**: Manual corrections or reclassifications applied directly in the GL due to reported system failure.

These differences are expected deviations, and identifying them requires domain judgment. A key challenge is distinguishing these expected deviations from true errors, such as missing transactions due to unreported system failures, which have to be corrected via additional manual adjustments.

### A.1. Typical Human Workflow

In practice, reconciliation is a multi-step process performed by human analysts:

1. **Data extraction.** Analysts retrieve GL and SL data, typically by writing SQL queries or requesting data from other teams via email or messaging tools.
2. **Workbook construction.** Analysts build an Excel workbook to compare GL and SL balances, aggregating data and computing differences.
3. **Deviation analysis.** Analysts investigate discrepancies to determine whether they are expected deviations (e.g., timing differences) or true errors (e.g., missing or incorrectly recorded transactions).
4. **Validation and reporting.** The final workbook serves as both a computational artifact and an audit record, often reviewed and approved by senior stakeholders.

### A.2. Role of Domain Knowledge

Beyond procedural steps, effective reconciliation relies heavily on domain knowledge about data patterns. Experienced analysts develop an internal understanding of:

- Seasonality (e.g., higher retail sales transaction volumes during sales and holidays, no bank transaction over weekends)
- Expected value ranges for different sources
- Typical behaviors of specific systems (e.g., known delays or processing quirks)

This knowledge enables them to quickly assess whether a deviation is expected or indicative of a real issue. For example, a small discrepancy at month-end may be normal due to timing, while an unusually large or persistent discrepancy may signal a system failure.

Such knowledge is rarely formalized in documentation but plays a critical role in decision-making. In this work, we treat this form of pattern-based reasoning as *domain knowledge*, and evaluate whether agents can acquire and leverage it to achieve senior-level understanding and judgment.

## B. On the Role of Golden Artifacts

A central component of our framework is the use of a "golden" artifact (e.g., a validated reconciliation workbook) as the starting point for replication-driven learning. We clarify the motivation for this design choice and its relevance to real-world enterprise settings.

### B.1. Golden Artifacts as Anchors for Learning

The proposed framework relies on reverse engineering, which fundamentally requires a reliable starting point. In noisy enterprise environments, knowledge is often distributed, incomplete, and internally inconsistent, making it difficult to determine correctness through aggregation alone.

In contrast to LLM pretraining - where correctness is implicitly approximated through large-scale statistical patterns - enterprise settings do not provide such guarantees. For example, a particular procedure or rule may appear repeatedly across multiple versions of documentation, yet still be incorrect due to outdated practices or copy-paste propagation. As a result, frequency of occurrence does not imply correctness.

Golden artifacts provide a necessary anchor of correctness in this setting. By definition, these artifacts represent outputs that have been validated and approved (e.g., signed off for audit or reporting purposes). They serve as a reference point against which candidate hypotheses can be evaluated during replication. Without such an anchor, the agent would have no principled way to distinguish between competing or conflicting knowledge sources.

### B.2. Practical Availability in Enterprise Workflows

The assumption of access to golden artifacts is not only methodologically useful but also practically realistic.

In enterprise environments, it is common to encounter multiple versions of documentation with inconsistent or outdated information, several intermediate or draft artifacts reflecting work-in-progress states, or a small number of finalized artifacts

that have been reviewed, approved, and used for reporting or auditing.

While curating and consolidating documentation requires significant effort and organizational alignment, sharing finalized artifacts is comparatively low-cost. In practice, stakeholders are often willing to provide validated outputs (e.g., a reconciliation workbook submitted for audit), even when comprehensive documentation is unavailable or unreliable. **In other words, it is often infeasible to enforce high-quality documentation at scale, but feasible to obtain validated artifacts.**

### B.3. Implications for Learning

The use of golden artifacts enables a learning paradigm based on working backwards from validated outputs, rather than relying on forward reasoning from potentially unreliable inputs. This has two key implications:

**Grounded verification.** Candidate procedures can be validated against a known-correct outcome, enabling iterative refinement through hypothesis testing.

**Robustness to noise.** Since correctness is anchored in the artifact rather than inferred from source frequency or consensus, the agent is less susceptible to systematic errors in distributed knowledge sources.

Importantly, golden artifacts do not contain complete procedural information. Key elements such as data extraction logic, filtering rules, and implicit assumptions may be missing, hard-coded, or indirectly encoded. As a result, replication still requires substantial reasoning and inference, ensuring that the learning process remains non-trivial.

## C. Example of Noise Simulation Config

To illustrate how knowledge is distributed and corrupted in the environment, we provide a simplified example of our noise distribution config file below with tables 2, 3, 4.

*Table 2.* Knowledge distribution across sources. Legend: ✓ = correct, × = incorrect, — = missing, N/A = not applicable.

| | Chunk | Doc V1 | Doc V2 | Kelly | ... | Bob | SQL History |
|---|---|---|---|---|---|---|---|
| 1 | GL Query | × | ✓ | — | ... | × | ✓ |
| 2 | SL Query (BankDeposit) | — | ✓ | — | ... | × | ✓ |
| 3 | SL Query (Remittance) | × | — | ✓ | ... | — | ✓ |
| 4 | ... | ... | ... | ... | ... | ... | ... |
| 5 | Reconciliation Formula | × | — | ✓ | ... | — | N/A |

*Table 3.* Examples of incorrect knowledge injected into sources.

| | Location | Chunk | Error Description | Wrong Value | Correct Value |
|---|---|---|---|---|---|
| 1 | Doc V1 | GL Query | Missing date filter | `SELECT...WHERE DESCRIPTION LIKE...` | `SELECT...WHERE PERIOD =...AND DESCRIPTION LIKE...` |
| 2 | Bob | GL Query | Uses wrong table (`gl_transactions` instead of `gl_entries`) | `SELECT...FROM gl_transactions` | `SELECT...FROM gl_entries` |
| 3 | Kelly | Adjustment Process | Incorrect identification logic | filter on keyword "adjustment" | filter on keyword "ADJ" |
| 4 | ... | ... | ... | ... | ... |

*Table 4.* Example SQL query history entries.

| | Query ID | Description | Query |
|---|---|---|---|
| 1 | d205... | GL extraction query | SELECT... |
| 2 | 6cd5... | SL query (BankDeposit) | SELECT... |
| 3 | e96b... | SL query (Remittance) | SELECT... |
| 4 | ... | ... | ... |

## D. Knowledge Distillation Prompt

### D.1. Stage 1: Learning Prompt

```
Learn from the {source_period_display} recon file and produce
distilled knowledge for future use.

Key Concepts:
- Golden source: The Excel recon file in {source_folder}/ is
   verified and correct
- Noisy environment: Documentation and people's knowledge may
   be outdated - reason through uncertainty using claims
- claims.md: Track hypotheses with evidence and status
   (verified/rejected/uncertain)
- knowledge.md & procedure.md: Created AFTER successful
   replication, containing only verified learnings

Constraints:
- Only read/write files within this workspace (no ../ paths)
- Excel sheets and CSVs may contain thousands of rows. Review
   wisely by checking file size, row count, header, review
   aggregated view.

Process:

Phase 1: Analysis with Sub-Agents
Launch two sub-agents to analyze different aspects of the
reconciliation:

Sub-Agent 1: Data Pattern Analyzer (CSV Focus)
Goal: Explore and document data patterns from CSV/SQL sources
into {target_folder}/claims.md
- Cross-reference with documentation and domain experts (Message/Email) to verify patterns
- Flag any discrepancies between data and other sources

**Claim format for data patterns**:
```
## Claim N: [Data Pattern Title]
**Claim**: [Your hypothesis about the data pattern]
**Source**: [SQL query / CSV file / person's name]
**Evidence**: [Query results, row counts, sample data that supports this]
**Status**: verified | rejected | uncertain
```
Sub-Agent 2: Excel Formula Analyzer (Excel Focus)
Goal: Reverse engineer the golden Excel file and draft the
recon script
- Study the {source_period_display} Excel workbook structure
- Document findings as claims in `{target_folder}/claims.md` (append to existing)
- Draft a Python script (`run_procedure.py`) that can recreate the workbook structure
- The script must use Excel formulas for calculations

**Claim format for Excel structure**:
```
## Claim N: [Excel Structure Title]
```
```

```
**Claim**: [Your hypothesis about what you observed]
**Source**: [Sheet name / cell reference / formula]
**Evidence**: [What you observed - exact formula, cell location, computed result]
**Status**: verified | rejected | uncertain
```

```
Phase 2: Merge and Review
After sub-agents complete:
- Review claims.md for completeness and consistency
- Ensure data patterns align with Excel formulas
- Resolve any conflicts between sub-agent findings
- Finalize the recon script based on combined understanding

Phase 3: Replication Testing
Run the script to generate the {source_period_display} recon
file and validate against the golden source. This is an
iterative process.

Phase 4: Document Verified Knowledge
After successful replication:
- Create {target_folder}/procedure.md
- Create {target_folder}/knowledge.md

Deliverables:
- claims.md, procedure.md, knowledge.md
- run_procedure.py
- summary.md
- test_recon.xlsx and validation files (csv/png/pdf)
```

### D.2. Variation of Stage 1 Learning Prompt

For ablation studies, we make the following modifications:

1. **No replication**: remove Phase 3 and deliverables `test_recon_*.csv/png/pdf`.
2. **No claims**: remove all wording related to claims, leaving only `procedure.md` and `knowledge.md`.
3. **Single agent**: remove wording "launch sub-agent" and phase 2 merge and review. We validate the logs to confirm the number of agents used.

### D.3. Stage 2: Task Execution Prompt

```
Generate the {target_period_display} recon file using the
distilled knowledge.

Available Resources: {available_resources}

Key Concepts:
- Prior knowledge: The md files and scripts were created by
  your colleague that learned from a verified golden source
- Scripts are tools: You CAN run the provided scripts if they
  exist - adapt the logic for current period as needed
- Knowledge as guardrail: Use procedure.md and knowledge.md
  to validate your work
- Noisy environment: Documentation and people's knowledge may
  be outdated - reason through uncertainty

Process:
Step 1: Review Available Resources in the folder
Step 2: Generate Target Period Reconciliation
  Option A: Use Existing Scripts (Recommended if available)
    Run scripts with {period} parameters
    Export key sheets to csv/png/PDF to validate calculated values against each documented data pattern

  Option B: Manual Generation
    Review prior month's recon files and documentation
```

```
    Use SQL/Message/Email to gather {target_period_display} data and knowledge
    Create Excel workbook with structure and format that exactly follows prior month's recon file
    Use Excel formulas for calculations
    Export key sheets to csv/png/PDF to validate calculated values. Document your findings. Modify the s
Step 3: note steps, key findings, results in summary.md

Deliverables:
- month_end_recon_{unit_id}_{target_period}.xlsx
- Validation files (csv/png/pdf)
- summary.md
```

## E. Excel Format Variants and Screenshots

*Table 5.* Excel workbook variants used in the simulated environments.

| Variant | Description |
|---|---|
| **Type A (Pivot)** | Reconciliation summaries use Excel pivot tables rather than explicit formulas. Agents must infer the underlying aggregation logic to replicate the calculation for new periods. |
| **Type B (Screenshot)** | One or more SL data sources exist only as embedded screenshot images rather than tabular data, simulating scenarios where source system access is restricted and data must be obtained through informal channels. Agents must identify the data owner from the image (not available anywhere else) and request updated data via email or messaging app. |
| **Type C (Hardcoded)** | Adjustment amounts are entered as literal values rather than formula references. Agents must distinguish formula cells from constants and reverse engineer the source and calculation logics of hardcoded values. |
| **Type D (Comment)** | Adjustment explanations and source references are stored in cell comments rather than visible cells. Agents must discover and extract metadata embedded in comment annotations. |

## F. Sample Q&A Questions

Q&A tasks are dynamically generated based on the actual data pattern per environment. For example, simulation environment 1 contains data from Japan subscription business whose reconciliation involves 5 SL data source and 1 GL data source. Questions and ground truth answers are then generated based on those 6 data sources and adapt to their unique data patterns. For each data source, we systematcially generate questions related to data extraction (columns, filter, groupby), reconciliation logic (field mapping between SL and GL), expected data pattern.

*Table 6.* Sample Q&A evaluation questions.

| Question | Ground Truth Answer |
|---|---|
| In GL, how should I get the data for SL Remittance? | Look for `journal_source = 'Remittance'` in GL entries. |
| In SL BankDeposit, which amount column maps to functional amount in GL? | `settlement_amount`. |
| I noticed there are no BankDesposit transactions on Feb 21, 2026 (Saturday). Should I talk to IT to confirm if there is any data ingestion failure?? | No, this account typically receives no bank deposits on Saturday. |
| I noticed a significant gap (over 1MM) between SL BankDeposit and GL on Dec 15–17. Is that expected? | No. Timing differences only occur near month boundaries (first/last few days). |
| For remittance entries in Japan music subscription, I noticed GL is consistently higher than SL at the beginning of the month. Is that expected? | No. Due to timing differences, SL should typically be higher than GL, not the reverse. |

*Note:* For illustration only, not real finance practice.

*Table 7.* Ablation study results under high noise (50% missing and incorrect knowledge).

| Noise Level | Variation | Excel Score | Q&A Acc. | Accuracy | Formula Usage | Layout | Format |
|---|---|---|---|---|---|---|---|
| High | No Replication | 83.7 | 84.8 | 87.0 | 93.6 | 70.3 | 11.9 |
| High | No Claims | 88.6 | 87.2 | 90.9 | 94.9 | 88.9 | 30.2 |
| High | Single Agent Distillation | 87.1 | 87.3 | 89.5 | 95.0 | 87.5 | 20.6 |

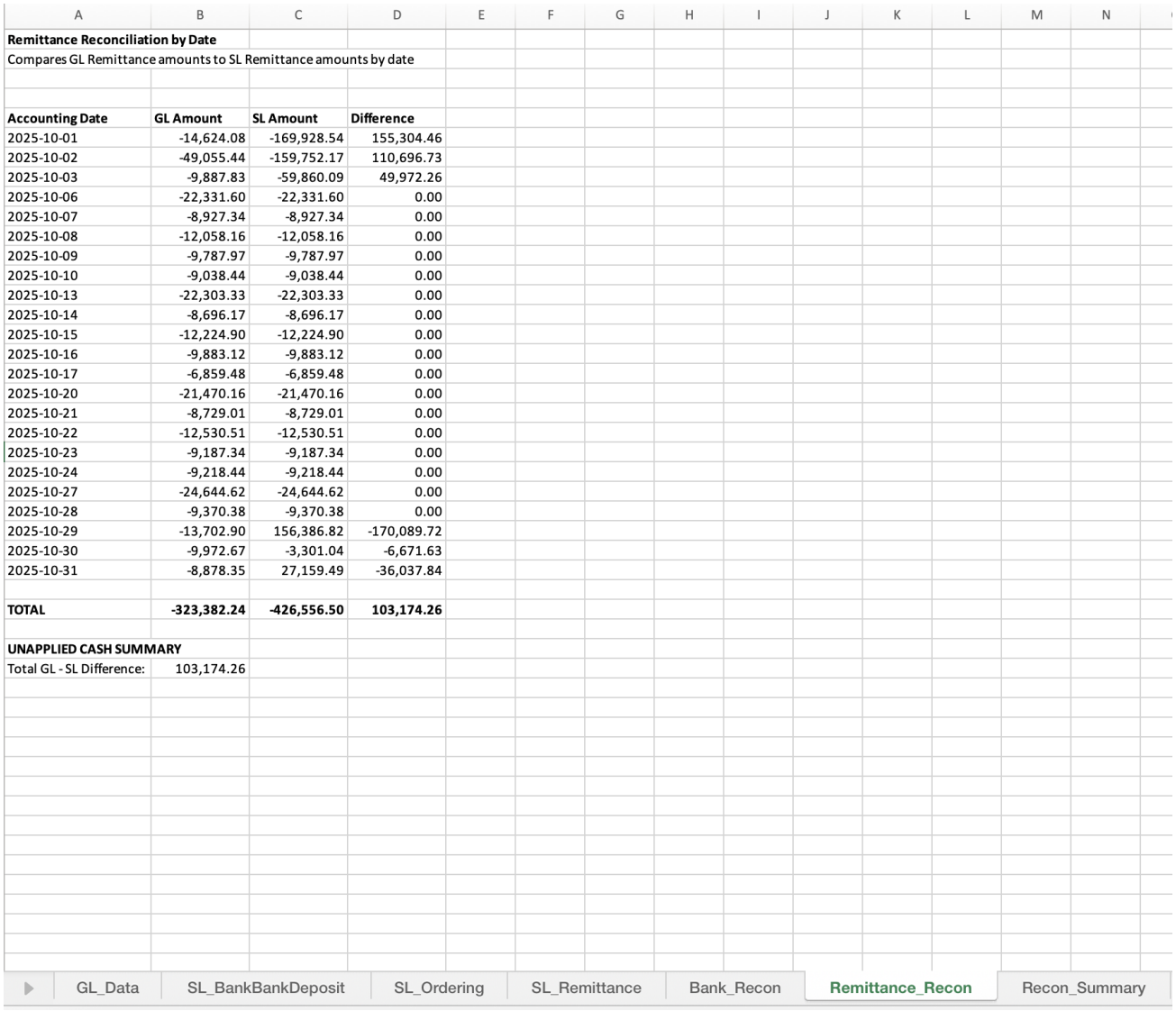
*Table 8.* Robustness under drift scenarios (high noise setting).

| Noise Level | Drift Scenario | Excel Score | Detection Rate | Accuracy | Formula Usage | Layout | Format |
|---|---|---|---|---|---|---|---|
| High | Schema Change | 88.2 | 98.3 | 90.0 | 99.2 | 90.8 | 15.8 |
| High | Aggregation Change | 90.3 | 88.3 | 92.8 | 99.5 | 90.9 | 17.5 |
| High | Ingestion Failure | 87.9 | 71.7 | 90.4 | 96.0 | 92.4 | 15.8 |

## G. Additional Evaluation Results

## H. LLM and Agent Usage Disclosure

The core ideas and problem formulation in this work are based on challenges observed in real-world production systems. The paper was initially drafted by the authors and subsequently refined and edited with the assistance of a SOTA LLM. The authors reviewed and validated all content for accuracy and originality.



**Remittance Reconciliation by Date**

Compares GL Remittance amounts to SL Remittance amounts by date

| Accounting Date | GL Amount | SL Amount | Difference |
|---|---|---|---|
| 2025-10-01 | -14,624.08 | -169,928.54 | 155,304.46 |
| 2025-10-02 | -49,055.44 | -159,752.17 | 110,696.73 |
| 2025-10-03 | -9,887.83 | -59,860.09 | 49,972.26 |
| 2025-10-06 | -22,331.60 | -22,331.60 | 0.00 |
| 2025-10-07 | -8,927.34 | -8,927.34 | 0.00 |
| 2025-10-08 | -12,058.16 | -12,058.16 | 0.00 |
| 2025-10-09 | -9,787.97 | -9,787.97 | 0.00 |
| 2025-10-10 | -9,038.44 | -9,038.44 | 0.00 |
| 2025-10-13 | -22,303.33 | -22,303.33 | 0.00 |
| 2025-10-14 | -8,696.17 | -8,696.17 | 0.00 |
| 2025-10-15 | -12,224.90 | -12,224.90 | 0.00 |
| 2025-10-16 | -9,883.12 | -9,883.12 | 0.00 |
| 2025-10-17 | -6,859.48 | -6,859.48 | 0.00 |
| 2025-10-20 | -21,470.16 | -21,470.16 | 0.00 |
| 2025-10-21 | -8,729.01 | -8,729.01 | 0.00 |
| 2025-10-22 | -12,530.51 | -12,530.51 | 0.00 |
| 2025-10-23 | -9,187.34 | -9,187.34 | 0.00 |
| 2025-10-24 | -9,218.44 | -9,218.44 | 0.00 |
| 2025-10-27 | -24,644.62 | -24,644.62 | 0.00 |
| 2025-10-28 | -9,370.38 | -9,370.38 | 0.00 |
| 2025-10-29 | -13,702.90 | 156,386.82 | -170,089.72 |
| 2025-10-30 | -9,972.67 | -3,301.04 | -6,671.63 |
| 2025-10-31 | -8,878.35 | 27,159.49 | -36,037.84 |
| **TOTAL** | **-323,382.24** | **-426,556.50** | **103,174.26** |

**UNAPPLIED CASH SUMMARY**

Total GL - SL Difference: 103,174.26

GL_Data | SL_BankBankDeposit | SL_Ordering | SL_Remittance | Bank_Recon | Remittance_Recon | Recon_Summary

*Figure 3.* Screenshot of Excel built by agent via replication. To build this sheet, agent needs to know the extraction logic of remittance data from GL, the extraction logic of remittance data from SL, date and amount field that matches in the two sources, then compare them date by date. For robustness scenarios, agent needs to know that expected deviation only occurs at the beginning and end of month and must be in the expected direction, positive at beginning of month, negative at end of the month.

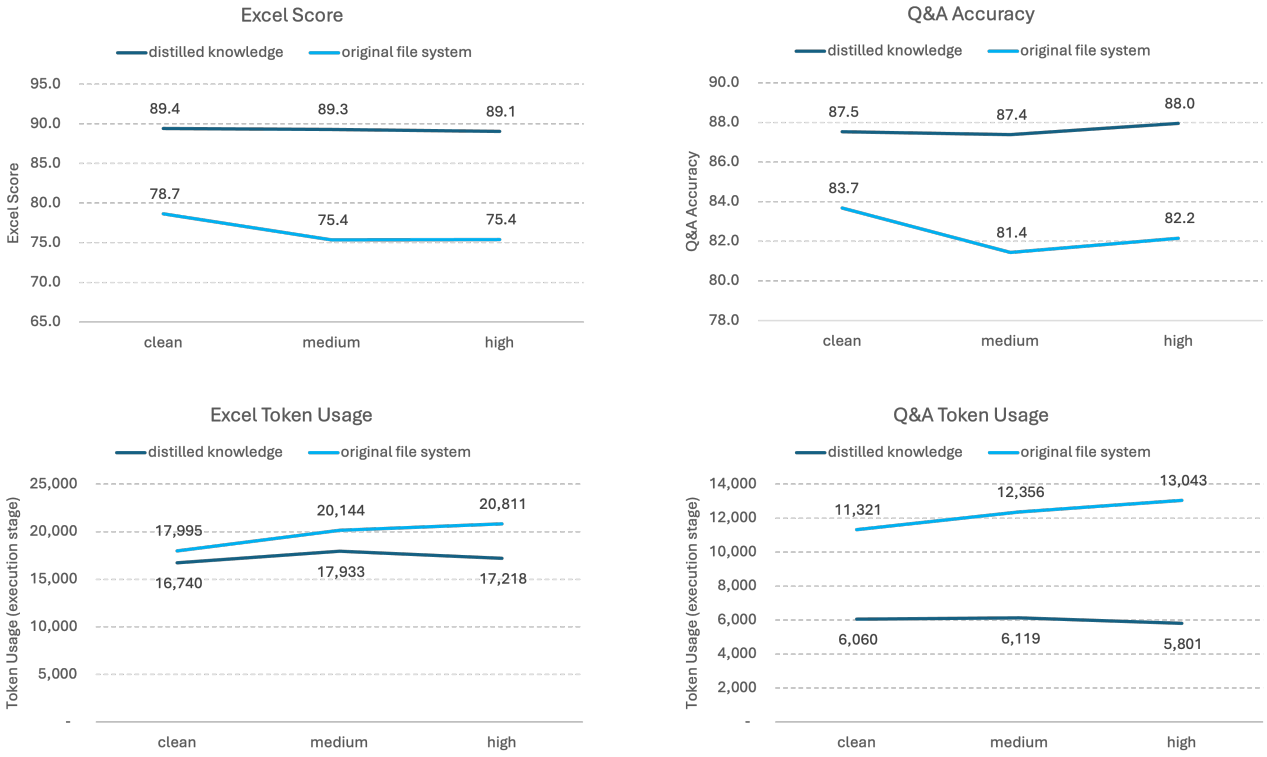



*Figure 4.* Chart of excel score, Q&A accuracy and token usage across 3 noise levels. Token usage reflects token used at execution stage only. The learning stage takes around 20-25K tokens on average. Clean, medium and high maps to 0%, 25% and 50% noise level