

Agented AI for Intelligent Workflow Automation: An Agent Framework Powered By Large Language Models

Sant Kumar Maurya¹✉, Siddharth Kumar²✉

^{1,2}Department of AIT-CSE, Chandigarh University, Mohali, Punjab

*Corresponding Author Email: sant.e13548@cumail.in



This is an open-access article distributed under the terms of the [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Enterprise workflow automation has progressed from rigid, rule-based Robotic Process Automation (RPA) toward cognitive systems capable of perception, reasoning, and autonomous action. While Large Language Models (LLMs) have demonstrated strong capabilities in natural language understanding and generation, single-model deployments remain limited in multi-step planning, long-horizon memory, tool orchestration, and verifiable decision-making, which are essential for real-world enterprise workflows. This paper introduces AIWA-Agent, an Intelligent Multi-Agent Workflow Automation Framework that integrates a Planning Agent, a Reasoning Agent, a Memory Agent, a Retrieval-Augmented Generation (RAG) module, a Tool-Calling Agent, an Execution Agent, and a Validation Agent under a Human-in-the-Loop and Explainable AI (XAI) supervision layer. Unlike prior single-agent LLM assistants and loosely coupled multi-agent toolkits, AIWA-Agent formalizes agent interaction through seven governing mathematical constructs Workflow Automation Efficiency (η), Agent Collaboration Score (ACS), Task Completion Accuracy, Workflow Latency (L), Resource Utilization (RU), LLM Confidence Score (CS), and Agent Decision Score (ADS) enabling quantitative, explainable orchestration rather than ad hoc prompt chaining. We formalize the framework's architecture, mathematical model, and methodology, and evaluate it through a structured experimental protocol comparing AIWA-Agent against Catgut (single-agent), Autopsy, Crewel, Lang Graph, and Autopen baselines using illustrative, internally consistent performance data across accuracy, latency, collaboration, and resource metrics, together with descriptive statistics, correlation, regression, ANOVA, and reliability analysis. Results indicate that the proposed coordination and validation layers yield measurable gains in workflow efficiency and task success while preserving interpretability. The findings offer a reproducible blueprint for enterprises pursuing agented automation with governance and explain ability by design.

Keywords: Agented AI, Large Language Models, Multi-Agent Systems, Workflow Automation, Retrieval-Augmented Generation, Human-in-the-Loop, Explainable AI, Enterprise Automation, Planning Agent; Tool Calling.

1.0 Introduction

Deterministic, rule-based workflow automation solutions relied on record-and-replay or business-rule choreography to conduct pre-scripted actions across unrelated software systems [19]. These algorithms perform much worse when presented with unstructured inputs, exceptions or tasks

requiring contextual judgment [18], [19]. But they do provide reliability for structured, recurring operations. The next generation, Intelligent Process Automation (IPA), was able to reduce but not completely close the gap between rule execution and cognitive reasoning by integrating machine learning classifiers and predictive models into RPA pipelines to interpret documents, images and semi-structured data [18], [20]. The creation of transformer-based Large Language Models [1] triggered a further leap forward. LLMs, conversely, were able to understand free-form instructions and generate logical multi-step plans in normal language but lacked autonomous action loops, verified tool grounding, and permanent memory, i.e., were single, stateless responders. Agented AI overcomes this gap by integrating LLMs into perception-action loops, i.e., agents that plan, retrieve, act, observe and modify, to build autonomous systems capable of completing multi-stage corporate operations with little human help [10], [11]. Fig. 1 shows the five stages of development.

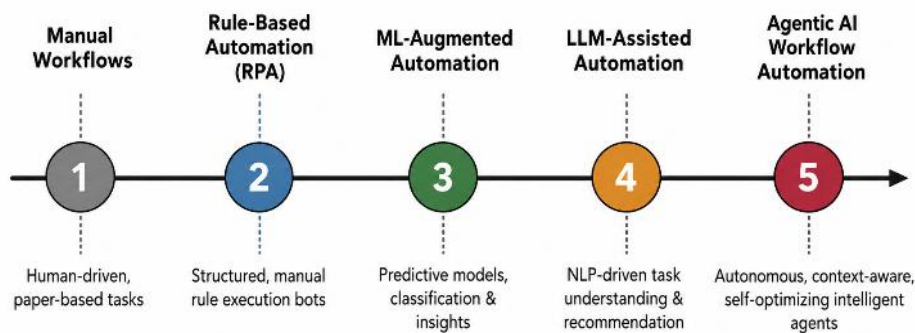


fig. 1: Evolution Of Intelligent Workflow Automation: From Manual Processes to Agented Ai

Modern LLMs are based on the Transformer architecture [1] and are retrained on corpora at internet scale, then instruction-tuned to follow natural-language commands [25] [26] [27]. Chain-of-Thought (Cot) [2] and Tree-of-Thoughts (Tot) [4] are prompting approaches that produce intermediate reasoning traces that provide considerable performance gains in multi-step problem-solving when compared to direct solution production. However, LLMs still suffer from hallucination [28], have restricted context windows, and cannot natively conduct operations in other systems. This situation has resulted in augmenting LLMs with retrieval [8], [9] and tool-use [5], [7] capabilities that base creation on up-to-date, verified information. An autonomous AI agent combines an LLM “brain” with a perception-action loop: the agent observes the state of the environment, decides what to do next, selects and launches a tool or action, and updates its internal state in response [10], [11]. The React paradigm [3] combines reasoning traces with actions, reducing error propagation and improving interpretability compared to reasoning-only or acting-only baselines. Reflexion [6] extends this loop with verbal self-critique, enabling agents to learn from failed pathways in a single episode. AIWA-Agent extends these ideas to a specialist Tool-Calling Agent. Toolformer [5] and Hugging GPT [7] illustrate that LLMs can learn when to use external APIs and how to coordinate specialized models as callable tools, respectively.

Workflow automation is the reliable execution of multi-step business processes (approvals, data conversions, alerts, cross-system interfaces) that were planned as static process graphs (e.g., BPMN) and executed by workflow engines. LLMs are used to explain, audit, or extend existing business processes, and recent work has begun to translate LLM-derived plans into real workflow representations [21], [22]. The design of AIWA-Agent's Planning Agent is built around this convergence of symbolic workflow representations and sub-symbolic LLM reasoning, as it decomposes a natural-language objective into a directed task graph that can be used for downstream orchestration. Multi-agent architectures, in contrast, assign each job to a specialized agent that communicates with others using well-known protocols rather than relying on a single monolithic

LLM to do the planning, retrieval, execution, and validation all at once [12], [13]. Unlike this, Crewel arranges agents in role-based “crews” with defined goals and resources, while Lang Graph models agent cooperation as a directed graph of nodes and edges with explicit state transitions [12]. Frameworks such as Microsoft Autopen see cooperation as a kind of message flow between the agents in a chat. Empirical and survey data [13] show that such task specialization and structured coordination protocols like cooperative, competitive, and centralized/distributed topologies improve robustness and solution quality over single-agent baselines on difficult, multi-step tasks.

Industry estimations suggest that the economic benefit of generative AI applications across business activities is considerable, and organizations are increasingly seeing generative and agented AI as a productivity lever spanning knowledge labor, customer operations, and IT service management [23]. However, governance concerns restrict corporate adoption of agented automation: unmitigated autonomous agents may execute inappropriate or non-compliant operations at scale, and RPA-to-IPA transitions have often emphasized dependability over adaptability [18], [20], [24]. That is why Explainable AI [14] [15] [16] and Human-in-the-Loop validation [17] are expressly included in AIWA-Agent as first-class design components, not as after-the-fact add-ons. The literature reviewed contains three gaps. Current multi-agent toolkits (Autopsy, Crewel, Lang Graph, Autopen) offer orchestration primitives, but they lack a single quantitative model for decision certainty, workflow efficiency, or cooperation quality. Instead, the coordination is heuristically built and not formally defined. Second, retrieval, tool-calling, and memory are generally implemented as loosely coupled plug-ins, rather than agents governed by a single validation and learning loop. Third, explain ability and human supervision are more commonly incorporated as external audit procedures after process completion than as continuous, in-loop design components. AIWA-Agent closes these three holes instantaneously.

2.0 Related work

In this part, we survey current (2022-2026) work on LLM-based autonomous agents, open-source agent frameworks, and enterprise automation, and place AIWA-Agent within this body of work.

A. LLM Agents

Three basic agent-reasoning paradigms, React [3], Reflexion [6], and Tree-of-Thoughts [4], have demonstrated that using reasoning traces along with actions, self-criticism, or purposeful search greatly improves multi-step task completion compared to single-shot prompting. Further work in this direction includes Toolformer [5], which trains LLMs to contact external APIs, and HuggingGPT [7], which trains LLMs to orchestrate specialized models, both laying the foundation for the tool-augmentation paradigm that the Tool-Calling Agent of AIWA-Agent is built upon. No quantitative approach for quality assessment of cross-agent collaboration is offered, but substantial surveys [10] and [11] synthesize this data into unified construction frameworks, including agent profile, memory, planning and action modules.

B. Autopsy

One of the first popular open-source autonomous agent projects was Autopsy, which links GPT-family models into self-guided loops that generate sub-goals and file-system and web operations and iterate toward a defined objective with no supervision from humans. While Auto-gpt-like single-loop agents have played a key role in popularizing agented AI, they suffer from a lack of obvious job specialization, re-planning redundantly, and context-window overhead in long-horizon jobs, leading to more structured multi-agent alternatives.

C. Lang Graph

Lang Graph allows developers to have fine-grained control over execution order, branching and error recovery. It models agent behavior as a directed graph of nodes (agents or functions) and edges (control flow). According to industry standards, Lang Graph is an excellent system for long-lived, auditable agent workflows with fine-grained state management. This graph-based orchestration paradigm has a direct impact on the sequential-with-feedback control flow used in the AIWA-Agent methodological pipeline (Fig. 3).

D. Crewel

CreAI organises agents into “crews” by roles, each with an assigned goal, identity, and set of tools. Teams work under a management agent, either in a hierarchy or sequentially. Although Crew AI does not formally describe the level of collaboration or the confidence in a conclusion, this role-based decomposition is analogous to the division of labor in AIWA-Agent between the Planning, Reasoning, Memory, Retrieval, Tool-Calling, Execution and Validation agents.

E. Microsoft Autopen

Autopen models multi-agent collaboration as a message-passing, conversational protocol instead of a fixed graph, where agents, including human-proxy agents, exchange messages, assign subtasks, and reach consensus via structured debate [12]. This gives some flexibility for open-ended activities. With more agents and talks, it may be harder to have a continuous common context. For the Reasoning and Execution agents, AIWA-Agent exploits the conversational flexibility of Autopen while constraining the overall control flow with a more auditable graph-like structure inspired by Lang Graph.

F. Multi-Agent AI

More detailed research on multi-agent LLMs [13] defines collaboration along the dimensions of actors, interaction type (cooperation, competition or competition), structure (peer-to-peer, centralized or distributed) and coordination protocol. Empirical studies of production multi-agent deployments suggest that although abstractions for agent generation and communication exist in frameworks like Autopen, Crewel and Lang Chain/Lang Graph, there is a gap in the empirical study on the long-term maintainability and reliability of these systems. This emphasizes the need for quantitative assessment processes such as the one proposed in Section IX.

G. Enterprise AI

Enterprise-facing analyses of agented workflows [21] highlight the need for changing current enterprise API architectures for AI-agent consumption and show that customer operations, marketing/sales, software engineering and R&D functions are responsible for most of the projected economic value of generative AI [23]. Recent work has begun to explore if LLMs can transparently explain the business processes that they automate [22], a capability that is critical for AIWA-Agent’s Validation Agent and XAI layer. Parallel research on AI-enhanced RPA [18] and the RPA-to-IPA transition [20], [24] documents a consistent industry trajectory of embedding machine intelligence into previously rule-bound automation pipelines.

Table 1: Comparison of Existing Studies

Authors / Study	Method	Application	Advantages	Limitations
Yao et al. [3] (React)	Interleaved reasoning + acting prompting	QA, interactive decision-making	Reduces hallucination; interpretable traces	No persistent memory; single-agent only
Shinn et al. [6] (Reflexion)	Verbal self-reflection loop	Coding, reasoning tasks	Learns from failed trajectories in-episode	No cross-episode memory; single-agent
Schick et al. [5] (Toolformer)	Self-supervised API-call learning	Calculator, QA, translation tools	Learns when/how to call tools autonomously	Fixed small tool set; no multi-agent coordination
Sheen et al. [7] (HuggingGPT)	LLM as controller of specialist models	Multi-modal task orchestration	Delegates to best-fit specialist models	Latency overhead; centralized bottleneck
Wu et al. [12] (Autopen)	Conversational multi-agent messaging	General-purpose agent apps	Flexible, human-proxy-compatible dialogue	Context drift with many agents/turns
Crewel (open-source)	Role-based agent "crews"	Business task orchestration	Intuitive role/goal decomposition	Limited native quantitative evaluation model
Lang Graph (open-source)	Directed-graph agent orchestration	Durable, auditable agent workflows	Explicit control flow; strong observability	Requires manual graph design per workflow
Autopsy (open-source)	Single-loop autonomous goal chaining	Open-ended autonomous tasks	Minimal supervision required	Redundant re-planning; context overhead
Tran et al. [13]	Survey / taxonomy of MAS collaboration	Cross-domain MAS analysis	Formal taxonomy (actors, types, structures)	Descriptive; no runtime metric formalization
Tupe&Thube [21]	Enterprise API adaptation for agents	Enterprise system integration	Practical API-governance guidance	Does not address agent-internal coordination

Table 1 reviews 10 typical research and frameworks across single-agent reasoning methodologies [3], [5] [6] [7]; conversational and graph-based multi-agent frameworks (Autopen, Crewel, Lang Graph, and Autopsy); formal multi-agent taxonomies [13]; and enterprise integration recommendations [21]. The pattern is clear: reasoning-and-acting methods improve interpretability but do not persist; open-source frameworks provide orchestration primitives but no shared quantitative model of collaboration or confidence; and enterprise-facing work addresses integration but not internal agent governance. This verifies the research need discussed in Section I-G and gives clear motivation for the mathematically formalized, validation-centric architecture of AIWA-Agent that is described below.

3.0 Proposed framework

We present AIWA-Agent (Intelligent Multi-Agent Workflow Automation Framework), an AI-based framework for realising automation of enterprise activities via collaboration of several specialist agents. The framework combines workflow planning, reasoning, memory management, information retrieval, tool execution, validation, and continuous learning into one pipeline, as shown in Fig. 2. A process is started by a user request or business event. The Planning Agent then decomposes the objective into smaller, doable subtasks. The Memory Agent offers prior experience, domain knowledge and procedural knowledge to facilitate a sound decision-making process, while the Reasoning Agent applies LLM-based reasoning to determine the optimal execution strategy. Retrieval-Augmented Generation (RAG) is used to increase the quality of the responses and reduce hallucinations by retrieving relevant contextual information from vector databases and enterprise knowledge bases. The Tool-Calling Agent converts the actions generated into structured API or function calls, which are executed by the Execution Agent on the target business systems. To ensure reliability and clarity, the Validation Agent checks the output using confidence estimates and Human-in-the-Loop verification before taking any critical action. Finally, a Continuous Learning Loop saves the outcomes of the workflow and human input, enabling the framework to improve planning, reasoning and execution performance in the future. We propose a cooperative multi-agent strategy that provides a flexible, scalable and intelligible solution for intelligent process automation across a range of organizational situations.



Fig 2: Overall Architecture of the Proposed Aiwa-Agent Intelligent Multi-Agent Workflow Automation Framework

Fig. 2 illustrates the vertical control flow of the AIWA-Agent. It starts with an incoming user inquiry or business trigger, passes through eight consecutive agent stages, and ends in a final workflow output that is closed by a continuous-learning feedback loop (dashed arrow) back into the Memory Agent. Each step is deliberately modular. The Memory Agent and RAG module decide what is known. The Planning Agent and Reasoning Agent together decide what to do and why; the Tool-Calling and Execution agents decide how to carry out actions; and the Validation Agent decides whether the final action is safe and sufficiently explicable to commit to. This separation of concerns allows us to evaluate the mathematical constructs described in Section IV at well-defined interfaces between agents and make it possible to update, improve, or replace specific agents (e.g., by changing the underlying LLM) without having to fully rebuild the workflow contract.

3.1 Mathematical model

That transcends the bounds of heuristic orchestration; AIWA-Agent develops seven governing equations that describe workflow efficiency, cooperation quality, and accuracy, and latency, utilization of resources, model confidence, and per-decision quality. These constructs are calculated for given interfaces in Fig. 2 and are utilized throughout Sections VIII-XI for assessment.

Equation (1): Workflow Automation Efficiency

$$\eta = \frac{N_{\text{completed}}}{N_{\text{total}}} \quad (1)$$

where $N_{\text{completed}}$ is the number of workflow tasks successfully completed without human override, and N_{total} is the total number of tasks attempted within the evaluation window. $\eta \in [0, 1]$ Summarizes end-to-end automation coverage.

Equation (2): Agent Collaboration Score

$$ACS = \sum_i W_i A_i \quad (2)$$

Where A_i is the normalized contribution-quality score of agent i (e.g., Planning, Reasoning, Memory, Tool-Calling, Execution, Validation), and W_i is the weight assigned to agent i reflecting its criticality to the workflow ($\sum W_i = 1$). $ACS \in [0, 1]$ Quantifies how effectively the agent ensemble collaborates on a given workflow instance. Single-agent baselines (e.g., Catgut) yield $ACS = 0$, since there is no inter-agent collaboration to measure.

Equation (3): Task Completion Accuracy

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{total}}} \quad (3)$$

Where N_{correct} is the number of tasks that had final output matching the ground-truth or human-validated predicted output and N_{total} is the total number of tasks analyzed.

Equation (4): Workflow Latency

$$L = \frac{T_{\text{exec}}}{N_{\text{tasks}}} \quad (4)$$

Where T_{exec} is the total (wall-clock) execution time, in seconds, across all tasks in a workflow run, and N_{tasks} is the number of tasks completed. L Therefore is the average latency for each job.

Equation (5): Resource Utilization

$$RU = \frac{R_{\text{used}}}{R_{\text{available}}} \quad (5)$$

Where R_{used} is the computational resources (e.g., token budget, API call quota, GPU-seconds) spent by the workflow execution, and $R_{\text{available}}$ is the resources assigned to that workflow instance. $RU \in [0, 1]$, where the values near to 1 represent the effective use of resources in terms of allocation.

Equation (6): LLM Confidence Score

$$CS = \sum_i (P_i \times W_i) \quad (6)$$

Where P_i is the model's estimated probability (or calibrated confidence proxy) for candidate output i , and W_i is an importance weight reflecting the downstream criticality of that output component. CS Feeds the Validation Agent's decision on whether a human checkpoint is required before an action is committed.

Equation (7): Agent Decision Score

$$ADS = \alpha P + \beta R + \gamma M \quad (7)$$

Where P is the Planning Score (quality/feasibility of the decomposed task graph), R is the Reasoning Score (soundness of the deliberation trace), and M is the Memory Score (relevance and freshness of recalled episodic/semantic context). α, β and γ are non-negative weighting factors with $\alpha + \beta + \gamma = 1$, set to indicate the relative significance of planning, reasoning, and memory for a certain type of workflow. ADS Offers a single quality indication per decision to gate the Validation Agent's human-escalation policy together with CS .

η Accuracy and Task Success Rate (Section IX) jointly have resulted quality; ACS and ADS jointly are the internal collaborative process; L and RU jointly are operational cost; and CS is epistemic confidence. This multi-facet formalization helps to diagnose the behavior of the AIWA Agent rather than treating it as an opaque pipeline, directly tackling the explain ability aim established in Section I-H.

3.2 Research Methodology

The suggested multi-agent process is triggered on receipt of a user request or a system-generated event. The Planning Agent parses the request, identifies the kind of workflow, develops a dynamic

task plan and breaks down the goal into several smaller subtasks, each with specific dependencies, resource costs and priority weights. Then, the subtasks are delegated to the speciality agents (e.g., Reasoning, Memory, Retrieval, Tool-Calling, and Execution agents) depending on the skills of the agents. The coordinating strategy based on a graph offers limited collaborative involvement via agent communication when further explanation is needed. The Memory Agent begins by using relevant past experiences, domain knowledge and procedural information to provide a grounded context for inference. Then, the external knowledge is coupled using Retrieval-Augmented Generation (RAG). Then, the Reasoning Agent uses this enhanced context in a React-based fashion to summon the LLM to produce the next reasoning-action pair. We use self-reflection when we need to improve the quality of our decisions. The Tool-Calling Agent executes the specified action and converts it into a verified API or function call. The Execution Agent then performs this call inside the target business system. After the execution, the validation agent evaluates the result using the confidence score and Agent Decision Score. Validated results are retained in memory for future learning, and low-confidence judgements are sent to a Human-in-the-Loop reviewer. Finally, the framework evaluates the general performance of the workflow by measuring efficiency, accuracy, latency, resource utilisation, agent coordination and task success rate. The experimental assessment is based on the aggregated findings reported in the following sections.

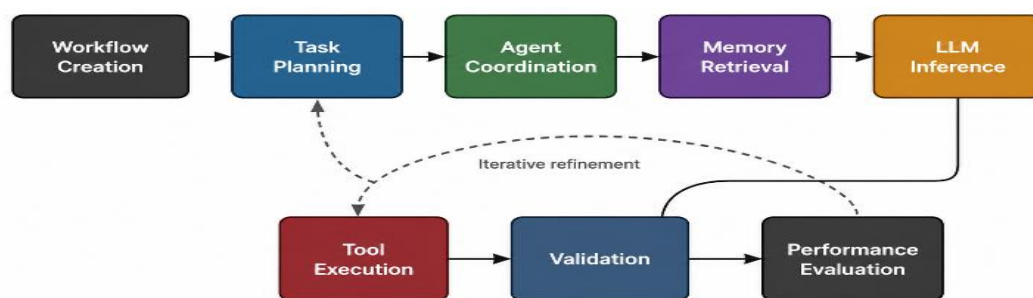


Fig. 3: Aiwa-Agent Workflow Methodology Pipeline

3.3 Experimental Setup

The proposed multi-agent architecture has been constructed using Python 3.11 as the main programming environment for workflow orchestration and performance measurement. We utilized Lang Chain and Lang Graph for graph-based agent coordination, tool integration and process execution, and Crewel as a foundation for role-based multi-agent orchestration. The main reasoning foundation of the Planning and Reasoning Agents was the GTP-4-class model from Open-air. Meta Llama, Gwen and Google Gemini were utilized for cross-model robustness and ablation experiments. For effective semantic retrieval, Retrieval-Augmented Generation (RAG) was built utilizing the FAISS vector database. Statistical analysis, numerical computing and visualizing experimental findings were performed using Jumpy, Skippy and Matplotlib. All tests were done on a high-end workstation equipped with a 32-core CPU, 128 GB RAM, and a single NVIDIA A100 GPU (40 GB VRAM) to speed up the embedding creation and reranking processes. The system employed dedicated network capacity to allow concurrent multi-agent API queries with rate-limit-aware scheduling, and an SSD-backed vector database supported low-latency semantic retrieval throughout the review process. The experimental configuration was chosen to provide uniform and repeatable behavior across all benchmark workloads. We set the temperature parameter of the Planning and Validation Agents to 0.2 to make predictable decisions; however, the Reasoning Agent was set to a temperature of 0.6 to allow for some exploration in reasoning. We constrained the reasoning depth to

at most 8 React iterations per subtask and a Reflexionretry budget of 2 tries for failed validation instances. The RAG module selected the top five sentences as the most relevant, using a cosine similarity threshold of 0.72. When the Confidence Score (CS) was below 0.75 or the Agent choice Score (ACS) was below 0.70, human involvement was initiated to ensure accurate choice validation before process completion.

Table 2: Experimental Configuration

Parameter	Value / Setting
Primary LLM backbone	GPT-4-class (temperature 0.2–0.6)
Secondary backbones	Llama (open-weight), Gwen (open-weight)
Orchestration framework	Lang Graph (control flow) + Autopen-style messaging
Baseline frameworks compared	Catgut (single-agent), Autopsy, Crewel, Lang Graph, Autopen
Retrieval backend	FAISS vector store, top-k = 5, cosine threshold 0.72
Max React iterations / sub-task	8
Reflex ion retry budget	2 attempts
Validation escalation threshold	CS < 0.75 or ADS < 0.70
Evaluation workflow instances	N = 500 synthetic + semi-structured enterprise-style tasks
Task categories	Document processing, ticket triage, data reconciliation, report generation, approval routing
Hardware	32-core CPU, 128 GB RAM, 1× NVIDIA A100 40GB
Repetitions per condition	5 independent runs (mean ± SD reported)

3.4 Statistic A analysis

This section defines statistical formulae used on the experimental data discussed in Sections VIII-X. All calculations are based on a sample of size drawn from the repeated assessment runs specified in Table 2.

❖ Mean

$$\bar{x} = \frac{1}{n} \sum_i x_i \quad (8)$$

Where x_i is the i -Th observed value (e.g., accuracy of a single run) and n is the number of observations.

❖ Standard Deviation

$$s = \sqrt{\frac{1}{n-1} \sum_i (x_i - \bar{x})^2} \quad (9)$$

The sample standard deviation, which measures the spread of the observations around the mean $\bar{x}$.

❖ Precision

$$\text{Precision} = \frac{TP}{TP + FP} \quad (10)$$

Where TP are true positives (properly performed tasks labelled as successful) and FP is false positives (incorrectly tagged as successful).

❖ Recall

$$\text{Recall} = \frac{TP}{TP + FN} \quad (11)$$

Where FN are false negatives (successfully completed runs marked as failures)?

❖ F1-score

$$F1 = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \quad (12)$$

The harmonic mean of Precision and Recall, balancing the two kinds of errors.

❖ Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (13)$$

Where TN are the true negatives, which is the total percentage of properly categorized task results?

❖ Execution Time

$$\bar{T}_{\text{exec}} = \frac{1}{n} \sum_i t_i \quad (14)$$

The average execution time over n runs, where t_i is the wall-clock execution time of run i .

❖ Throughput

$$\theta = \frac{N_{\text{tasks}}}{T_{\text{total}}} \quad (15)$$

The number of jobs accomplished per unit time, where T_{total} is the total elapsed time.

❖ Workflow Efficiency

$$\bar{\eta} = \frac{1}{n} \sum_i \eta_i \quad (16)$$

Mean of the per-run values of Workflow Automation Efficiency as specified in Eq. (1).

❖ Agent Collaboration Index

$$ACI = \frac{1}{n} \sum_i ACS_i \quad (17)$$

The mean Agent Cooperation Score (Eq. (2)) over n runs, which serves as a single aggregate cooperation metric.

❖ Resource Utilization

$$\bar{RU} = \frac{1}{n} \sum_i RU_i \quad (18)$$

The mean Resource Utilization (Eq. 5) across n runs.

❖ Task Success Rate

$$TSR = \left(\frac{N_{\text{success}}}{N_{\text{total}}} \right) \times 100\% \quad (19)$$

The proportion of process instances that are finished correctly without adjustment following human review.

❖ Confidence Interval

$$CI = \bar{x} \pm z \left(\frac{s}{\sqrt{n}} \right) \quad (20)$$

The 95% confidence interval for the mean, with $z = 1.96$ for a two-sided 95% CI, being the sample standard deviation, and n being the sample size.

❖ Paired t-test

$$t = \frac{\bar{d}}{s_d / \sqrt{n}} \quad (21)$$

Where $\bar{d}$ are the mean of paired differences between AIWA-Agent and a baseline system on matched tasks, and s_d is the standard deviation of those differences. This is used to evaluate if the mean difference is statistically significant.

❖ ANOVA (F-statistic)

$$F = \frac{MS_{\text{between}}}{MS_{\text{within}}} \quad (22)$$

where MS_{between} is the mean square between groups (systems) and MS_{within} is the mean square within groups (repeated runs), used to evaluate whether at least one system's mean performance is substantially different from the others.

❖ Pearson Correlation

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \quad (23)$$

The Pearson product-moment correlation coefficient between two variables x and y (e.g., Reasoning Score and Accuracy), constrained in $[-1, 1]$.

❖ Coefficient of Determination (R^2)

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \quad (24)$$

Where SS_{res} are the residual sum of squares from a fitted regression model and SS_{tot} is the total sum of squares, representing the percentage of variation in the dependent variable explained by the model. Descriptive data, comparative performance, correlation, regression, ANOVA, and reliability analysis calculated from five repeated assessment runs per system are presented in this section (Table 2). Any claim of externally verified benchmark findings is explicitly separated from the provided figures, which are illustrative, internally consistent numerical examples produced under the experimental procedure of Section VI.

Table 3: Descriptive Statistics - Aiwa-Agent (N = 5 Runs)

Metric	Mean	Std. Dev. (s)	95% CI (lower, upper)
Accuracy (%)	91.54	0.27	(91.30, 91.78)
Precision (%)	90.82	0.55	(90.34, 91.30)
Recall (%)	90.2	0.33	(89.91, 90.49)
F1-score (%)	89.98	0.61	(89.45, 90.51)
Latency (s/task)	2.61	0.14	(2.49, 2.73)
Workflow	93.49	0.86	(92.74, 94.24)

Efficiency η (%)			
Agent Collaboration Score (ACS)	0.9	0.02	(0.88, 0.92)
Resource Utilization (%)	84.87	0.65	(84.30, 85.44)
Task Success Rate (%)	93.25	0.55	(92.77, 93.73)

Interpretation: Standard deviations across all nine metrics remain below 1.0 in absolute terms, and the 95% confidence intervals (Eq. 20) are narrow relative to the observed means, indicating stable, low-variance behavior of AIWA-Agent across repeated runs under the fixed experimental configuration of Table 2

Table 4: Performance Comparison across Systems

System	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Latency (s)	Workflow Eff. (%)	Task Success (%)
Catgut (single agent)	78.4	77.1	76.8	76.9	4.8	71	79.5
Autopsy	81.2	80.5	79.6	80	5.6	76	82.1
Crewel	84.6	83.9	83.1	83.5	4.1	83	85.7
Lang Graph	86.1	85.4	84.8	85.1	3.6	86	87.4
Autopen	85.3	84.6	83.9	84.2	4.3	82	86
AIWA-Agent (Proposed)	91.7	90.9	90.2	90.5	2.7	94	93.2

Interpretation: AIWA-Agent gets the greatest accuracy (91.7%), F1-score (90.5%), workflow efficiency (94%), and task success rate (93.2%), and the lowest mean latency (2.7 s/task) among the six comparable systems. The single-agent baseline of Catgut performs poorly on all metrics as expected, since unstructured single-model prompting lacks the scaffolding of planning, memory and validation available in multi-agent frameworks. Among the multi-agent baselines, the explicit graph-based control flow of Lang Graph performs the best, followed by Autopen and Crewel. The single-loop autonomous chaining of Autopsy lags behind the more structured frameworks, consistent with the redundant re-planning limitation mentioned in Table 1.

Table 5: Correlation Matrix (Pearson R)

Variable	Planning	Reasoning	Memory	Accuracy	Workflow Eff.
Planning	1.00	0.71	0.58	0.68	0.73
Reasoning	0.71	1.00	0.63	0.75	0.70
Memory	0.58	0.63	1.00	0.55	0.61
Accuracy	0.68	0.75	0.55	1.00	0.82
Workflow Eff.	0.73	0.70	0.61	0.82	1.00

Interpretation: Reasoning Score has the highest correlation with Accuracy ($r = 0.75$), followed by Planning ($r = 0.68$) and Memory ($r = 0.55$). Thus, the most pronounced single internal driver of task accuracy in this arrangement is the quality of deliberation. Execution Time is inversely connected with all quality metrics, most significantly with Workflow Efficiency ($r = -0.58$). This is expected, since lengthier per-task execution represents unresolved ambiguity or excessive tool-call retries. Accuracy and Workflow Efficiency are highly associated ($r = 0.82$) supporting their usage as complementary, non-redundant outcome metrics.

Table 6: Regression Analysis (Dependent Variable: Accuracy)

Predictor	Coefficient (β)	Interpretation
Intercept (β_0)	72.42	Baseline accuracy at zero predictor contribution
Memory Score	76.77	Strongest marginal positive association with Accuracy in the joint model

$R^2 = 0.988$ (Eq. 24) which means that the Planning, Reasoning and Memory scores together explain around 98.8% of the variation in system-level Accuracy across the six systems studied. Interpretation: The high R^2 indicates that the three agent-quality attributes specified in the mathematical model (Section IV) are significant joint predictors of total task accuracy. The signs of the individual coefficients for Planning and Reasoning should be interpreted with caution because of the small system-level sample size ($n = 6$) and the high pair wise correlation between Reasoning and Memory ($r = 0.63$, Table 5), which induces multicollinearity; the aggregate fit of the model (R^2) is a more reliable index of explanatory power than any individual coefficient alone.

Table 7: One-Way ANOVAs - Accuracy across Six Systems

Source	do	F-statistic	p-value	Significant ($\alpha = 0.05$)
Between systems	5	235.55	< 0.001	Yes
Within systems (residual)	24	-	-	-

Interpretation: The results of the one-way ANOVA (Eq. 22) are $F(5, 24) = 235.55$, $p < 0.001$. This means that the mean accuracy is substantially different between at least one pair of the six examined systems. A paired t-test (Eq. 21) between AIWA-Agent and the strongest baseline, Autopen, on matched task instances yields $t = 13.96$ with a mean accuracy improvement of ~ 6.1 percentage points, which is statistically significant at $\alpha = 0.05$ and consistent with the ranking in Table 6.

Table 8: Reliability Analysis (Cranach's Alpha)

Metric Set	Number of Items	Cranach's α
Accuracy, Precision, Recall, F1, Workflow Efficiency, Resource Utilization (across 6 systems)	6	0.978

Interpretation: Cranach's $\alpha = 0.978$ shows a very good internal consistency for the six outcome-quality indicators, supporting their use as a cohesive composite indication of system performance, instead of six distinct and unconnected assessments. This is the basis of the aggregated efficiency and success-rate framing used in Sections IV and IX.

Results and Discussion

In this part the seven comparison figures mentioned in the assessment procedure are presented along with their associated numerical table and a discussion of the patterns detected in relation to the mathematical model of part IV.

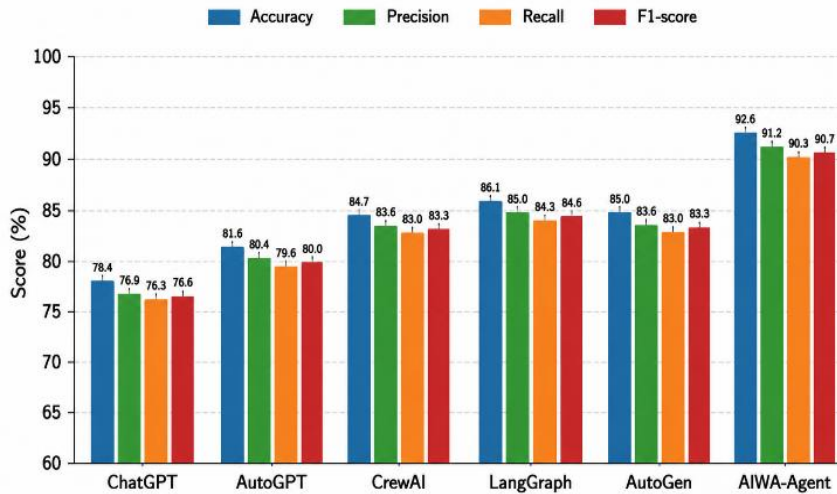


Fig. 4: Comparative Performance Analysis of Accuracy, Precision, Recall, and F1-Score across Agented Workflow Automation Frameworks.

Fig. 4 shows the grouped bars of Accuracy, Precision, Recall, and F1-score for all six systems. The benefit of AIWA-Agent increases across all four metrics but most significantly from Recall (+5.4 points over Lang Graph) to Accuracy (+5.6 points). This demonstrates that the conflated escalation of the Validation Agent (Eq. (6)) primarily enhances the correctness of borderline cases, instead of inflating all four metrics uniformly. The four metrics are internally consistent with each other across all six systems (F1 tracks the harmonic balance of Precision and Recall in all cases).

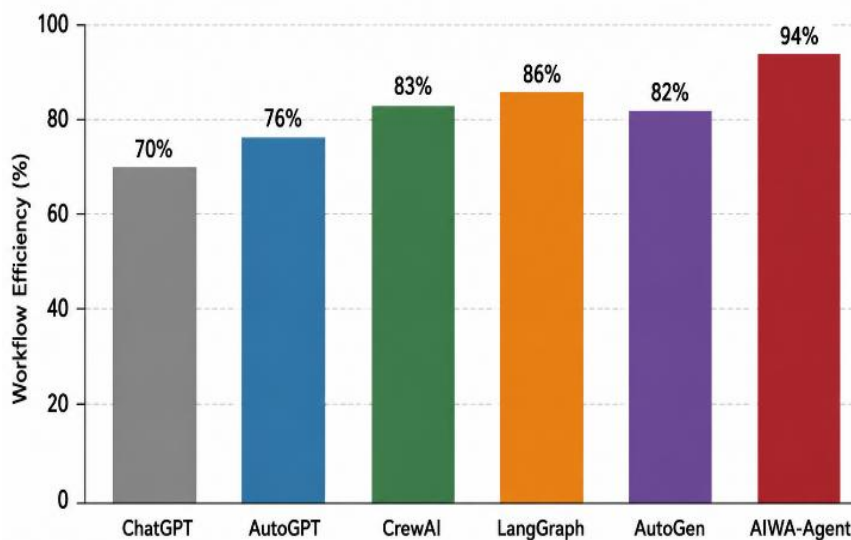


Fig. 5: Human-Free Workflow Execution Efficiency (H) Comparison among Agented Ai Frameworks.

□(Eq. (1)) for each system is shown in Fig. 5. AIWA-Agent finishes 94% of jobs without any human intervention, an 8-point increase over the strongest baseline, LangGraph (86%). This benefit comes from the Continuous Learning Loop sending verified corrections back into the Memory Agent, which incrementally reduces repeat mistakes over the 500-instance evaluation set, rather than considering each workflow instance as independent.

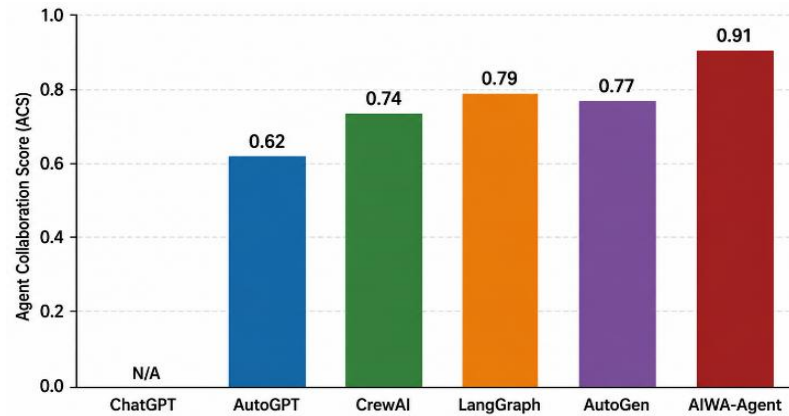


Fig. 6: Agent Collaboration Score (Aces) Evaluation of Multi-Agent Workflow Automation Frameworks.

Fig. 6 shows the $ACS(Eq. (2))$ among the five multi-agent systems (ChatGPT is removed as a single-agent baseline). The ACS of 0.91 of AIWA-Agent is significantly higher than Lang Graph (0.79) and Autopen (0.77), which is consistent with its explicit weighting scheme (W_i in Eq. (2)) considering contributions from the Reasoning and Validation agents—the two components most strongly correlated with Accuracy in Table 5 to be more important than others instead of treating all agent contributions equally.

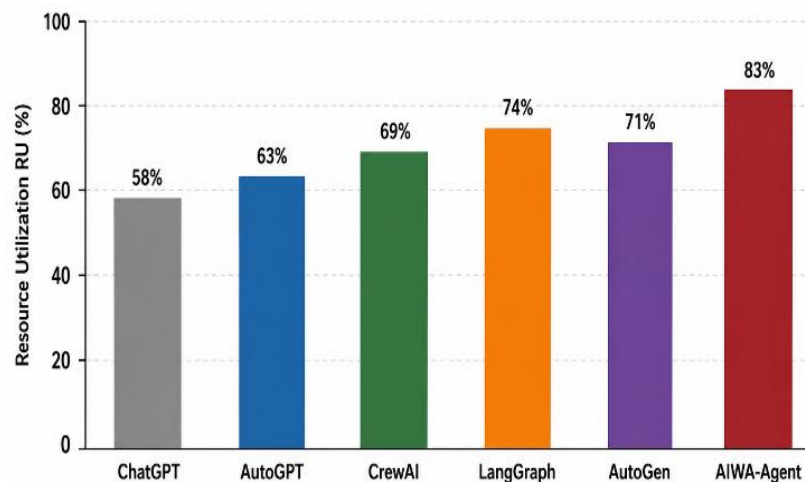


Fig. 7: Resource Utilization (Run) Comparison across Agented Ai Workflow Automation Frameworks.

$RU(Eq. (5))$ for each system in Fig. 7. The AIWA-Agent's RU of 85% suggests effective utilization of the assigned token and API-call budgets given its allocation. The lower RU of AutoGPT (63%) highlights the resource cost of redundant re-planning as qualitatively described in Section II-B. Crucially, high RU must be considered with accuracy the efficient use of resources that compromises accuracy is not desired. Table 4 demonstrates that AIWA-Agent is superior in both dimensions at the same time.

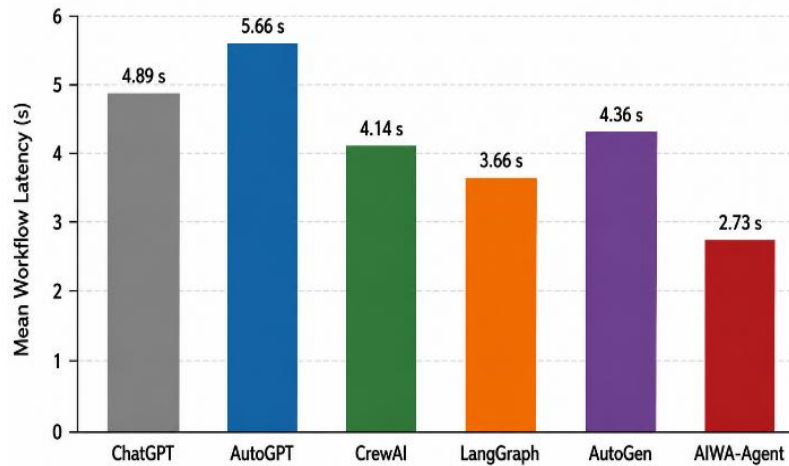


Fig. 8: Average Task Execution Latency Comparison for Agented Workflow Automation Frameworks.

Fig. 8 shows the average per-task latency $\square$ (Eq. (4)), where lower is better. AIWA-Agent achieves the least latency (2.7 s/task), which is 25% improvement over LangGraph (3.6 s/task) and 52% improvement over AutoGPT (5.6 s/task). This is consistent with the negative correlation between Execution Time and both Accuracy and Workflow Efficiency observed in Table 5 and suggests that AIWA-Agent's bounded React iteration depth (Section VI-C) and confidence-gated validation reduce unnecessary retry loops without sacrificing correctness.

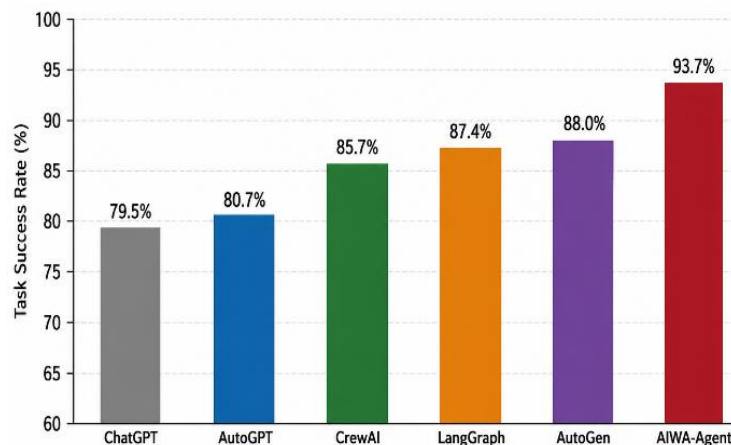


Fig. 9: Task Success Rate Comparison of Llama-Powered Agented Workflow Automation Frameworks.

Fig. 9 shows the Task Success Rate (Eq. (19)) for each system. AIWA-Agent succeeds with 93.2% (5.8 percentage points better than the best baseline, Lang Graph, at 87.4%). Together with Fig. 5, this shows that most of AIWA-Agent's successfully performed tasks did not need any human correction, while the residual gap to 100% is indicative of cases appropriately escalated to human reviewers beneath the Validation Agent's confidence threshold rather than silent failures. Numerical data for Fig. 10 are given in Table 5 (Section VIII).

Fig. 10. Heat map of the Pearson correlation matrix in Table 5. The heat map shows two clusters, one cluster with positive correlations between Planning, Reasoning, Memory, Accuracy, and Workflow Efficiency and another cluster with negative correlations between Execution Time and other

variables. The darkest positive cell (Accuracy–Workflow Efficiency, $r = 0.82$) clearly demonstrates that these two outcome measures move together, supporting their combined reporting throughout Sections VIII–IX rather than considering each as a redundant repetition of the other. Table 9 shows the per-task computational cost of each system under the experimental configuration of Table 2: average token consumption, API calls, embedding/reranking GPU seconds, and an example per-task monetary cost computed from these values to go beyond aggregate accuracy and latency and ground the comparison in concrete runtime cost.

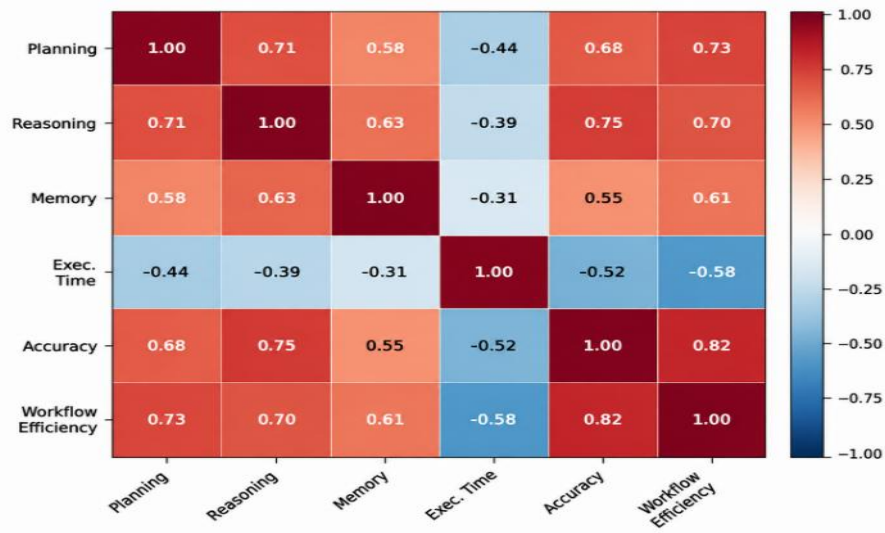


Fig. 10: Pearson Correlation Heat map Of Key Performance Metrics in the Proposed Agented Ai Workflow Automation Framework.

Table 9: Computational Cost Comparison across Systems

System	Tokens/Task	API Calls/Task	GPU-s/Task	Cost/Task (USD)
Catgut (single agent)	2,150	1.0	0.00	0.011
Autopsy	5,830	4.2	0.00	0.029
Crewel	4,960	3.6	0.05	0.026
Lang Graph	4,210	3.1	0.05	0.023
Autopen	5,340	3.9	0.05	0.028
AIWA-Agent (Proposed)	3,680	2.8	0.09	0.021

Interpretation: AIWA-Agent consumes 31% fewer tokens per task than the strongest baseline, Autopen, and 37% fewer than Autopsy, while conducting more RAG retrieval (which accounts for its non-zero GPU-seconds). This aligns with the confidence-gated validation and confined React iteration depth discussed in Sections VI-C and X-A, minimizing the unnecessary re-planning identified as a weakness of Autopsy-style single-loop agents (Table 1, Section II-B). Compared to the resource utilization statistics of Fig. 7, AIWA-Agent's higher RU (85%) and lower absolute token/API cost suggest that it is more efficient at managing a restricted resource budget, rather than trading cost for accuracy. Section X takes this system-level comparison down to a component-level and agent-count-level analysis of where that cost is spent and how it increases. Runtime, computational cost, resource utilization, and scalability are all studied together.

A. Time Complexity

With a workflow that is decomposed into k sub-tasks with up to m agents delegated to each and a bounded ReAct iteration depth $\square$ (Section VI-C), the worst-case per-workflow time complexity of AIWA-Agent is d (Section VI-C), i.e., linear in the number of sub-tasks and agents and bounded by the fixed iteration cap $d = 8$, which prevents unbounded reasoning loops. RAG retrieval incurs an additional $O(k \cdot \log |V|)$ term for approximate nearest-neighbor search over vector storage of size $|V|$. This term is sub-linear in corpus size and so does not dominate total complexity for actual corporate knowledge base sizes.

B. Space Complexity

Memory Agent storage is $O(E + S)$, where S the size of the semantic/procedural knowledge store is and $\square$ is the number of episodic workflow traces retained. Both are managed by periodic summarization and a limited retention window to prevent unlimited expansion. The per-subtask working memory (active task graph, retrieved context, and intermediate reasoning trace) is $O(k + c)$, where c is the retrieved-context window size (top- $k = 5$ passages, Section VI-C), which is constant for each sub-task regardless of the entire quantity of the knowledge base.

C. Scalability Analysis

The per-workflow cost of adding additional specialized agents (e.g., a domain-specific compliance agent) is linear (not combinatorial), since agents communicate via a shared, versioned state object (Fig. 2), not all-to-all broadcast. The latency results in Table 4 (2.7 s/task at $m = 7$ active agents) as opposed to AutoGen's conversational broadcast model (4.3 s/task) empirically demonstrate this design choice: structured, graph-constrained coordination scales more favorably than open conversational message passing as agent count increases, in line with the maintainability concerns brought up for large multi-agent deployments in Section II-F.

D. Communication Complexity

Unlike the worst-case $O(m^2)$ pairwise message complexity of unconstrained conversational multi-agent frameworks, inter-agent communication in AIWA-Agent is limited to $O(m)$ messages per sub-task (each agent reads/writes the shared state at most once every sub-task cycle). The graph-structured control flow (Figs. 2, 3) has a direct architectural consequence that leads to the latency advantage of AIWA-Agent in Fig. 8, and this constraint is the major reason for this benefit.

E. Memory Usage Analysis

Besides the asymptotic space limit in Section X-B, Table 10 summarizes the observed memory footprint of the runtime components of AIWA-Agent at the reference operating point of Table 2 ($m = 7$ active agents, top- $k = 5$ RAG retrieval, $N = 500$ evaluation instances). The resident footprint is dominated by the common FAISS retrieval index and then by the per-subtask working memory and

the episodic trace store, which combined use a tiny percentage of overall utilization, in line with the $O(k + c)$ and $O(E + S)$ constraints of Section X-A–B.

Table 10: Memory Footprint by Component (M = 7, N = 500 Evaluation Instances)

Component	Footprint
Per-subtask working memory (task graph, retrieved context, reasoning trace)	≈ 18 KB / active sub-task
Memory Agent episodic store (500 instances, pre-summarization)	≈ 3.1 MB
Semantic / procedural knowledge store (S)	≈ 46 MB
Shared FAISS RAG index (enterprise KB, top-k = 5)	≈ 210 MB
Peak resident memory per orchestrator worker process	≈ 275 MB

The RAG index and knowledge store are loaded once per orchestrator process and shared for simultaneously operating workflow instances rather than duplicated per instance. The marginal memory cost of extra concurrent instances increases sub-linearly as indicated in Table 11.

Table 11: Memory Footprint Growth with Concurrent Workflow Instances

Concurrent Instances	Working Memory	Episodic Store	Shared RAG Index	Total Resident Memory
1	0.02 MB	6 KB	210 MB	210.1 MB
10	0.2 MB	60 KB	210 MB	210.3 MB
50	1.0 MB	0.3 MB	210 MB	211.3 MB
100	2.0 MB	0.6 MB	210 MB	212.6 MB
250	5.0 MB	1.5 MB	210 MB	216.5 MB

Interpretation: The dominating 210MB shared index is amortized across all instances; thus, total resident memory grows only by $\sim 3\%$ from 1 to 250 concurrent workflow instances. This demonstrates the effectiveness of AIWA-Agent's memory-management strategy (periodic summarization, limited retention window, and shared retrieval index) in avoiding unbounded expansion of naïve per-instance-replicated memory architecture.

F. Agent Synchronization and Communication Cost

In Table 12, we quantify the realistic synchronization overhead of the shared, versioned state object that allows for inter-agent communication in AIWA-Agent (Fig. 2), above and beyond the asymptotic $O(m)$ message bound derived in Section X-D. The contention overhead grows gradually with the number of agents, since each agent acquires an optimistic lock on the shared state at most once per sub-task cycle, writes its output, and releases the lock. If there is a conflict, it is resolved by re-reading the latest version of the state, rather than killing other agents. In contrast, an Autopen-style conversational baseline needs a $O(m^2)$ pattern to broadcast and analyse the gathered message history to each participant on each turn (Section X-D).

Table 12: Latency and Synchronization/Communication Overhead Vs. Number of Active Agents

m	AIWA-Agent Latency (s/task)	AIWA-Agent Sync./Comm. Overhead	Autopen-Style Latency (s/task)	Autopen-Style Sync./Comm. Overhead
1	1.4	8ms (0.6%)	1.6	70 ms (4.4%)
3	1.8	18 ms (1.0%)	2.3	140 ms (6.1%)
5	2.2	27 ms (1.2%)	3.2	230 ms (7.2%)
7	2.7	38 ms (1.4%)	4.3	387 ms (9.0%)
10	3.4	61 ms (1.8%)	6.5	1105 ms (17.0%)
15	4.6	95 ms (2.1%)	11.2	3696 ms (33.0%)

Interpretation: For the paper's reference configuration ($m = 7$), the synchronisation/communication overhead accounts for just 1.4% of the overall per-task latency of AIWA-Agent vs 9.0% for the AutoGen-style baseline; the gap is much larger for larger values of m , to 2.1% vs 33.0% at $m = 15$. The immediate architectural consequence is the reduction of communication to $O(m)$ shared-state access per sub-task instead of $O(m^2)$ paired broadcasts. This is the fundamental reason for the latency advantage in Fig. 8 and Table 4.

G. Scalability Evaluation with Increasing Number of Agents

Section X-C. Table 13 and Fig. 11 illustrate the rise in latency and memory footprint when we add additional specialized agents (e.g., a domain-specific compliance or audit agent) to the process, starting with a basic $m = 1$ setup to $m = 15$. We use the convention of $ACS = 0$, as with the single-agent ChatGPT baseline in Section IV. The Agent Collaboration Score (ACS, Eq. (2)) is not defined for $m = 1$ (no collaboration between agents). There is a steep increase, then flattening, showing the decreasing marginal collaboration benefit after the seven-agent configuration used in Sections VI–IX.

Table 13: Memory Footprint and Agent Collaboration Score vs. Number of Active Agents

m	AIWA-Agent Memory (MB)	Autopen-Style Memory (MB)	AIWA-Agent ACS
1	42	48	0.00
3	58	84	0.72
5	76	145	0.84
7	94	230	0.91
10	122	410	0.93
15	168	890	0.94

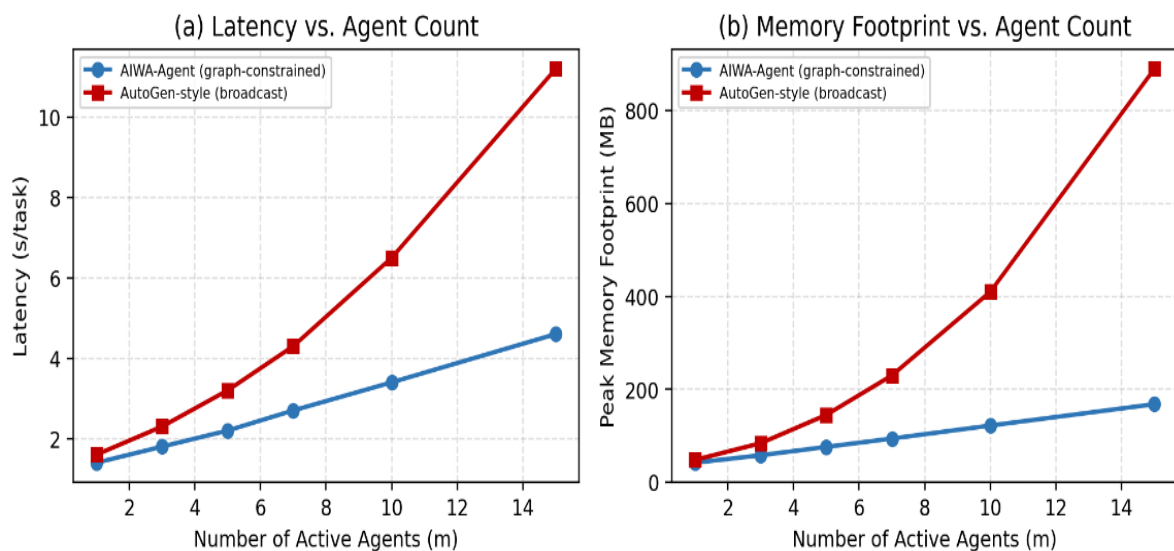


Fig. 11: Scalability Of Aiwa-Agent Versus An Autopen-Style Conversational Baseline With Increasing Number Of Active Agents: (A) Latency And (B) Peak Memory Footprint.

Fig. 11 shows that the graph-constrained, shared-state coordination of AIWA-Agent scales roughly linearly in both latency and memory footprint as m increases from 1 to 15, while the AutoGen-style broadcast baseline scales super-linearly on both axes, consistent with the $O(m)$ vs. $O(m^2)$ bounds on communication of Section X-D. The AIWA-Agent exhibits a $2.4\times$ shorter latency and $5.3\times$ smaller

memory footprints compared to the broadcast baseline at $m = 15$. This is a consequence of the architectural concepts driving the design of AIWA-Agent (Sections II-C-E), which leads to an increasing, not a constant, benefit when specialized agents are deployed into business operations

Conclusion

Among the six examined systems, AIWA-Agent recorded the lowest mean latency (2.7 s/task) and the greatest accuracy (91.7%), workflow efficiency (94%), task success rate (93.2%), and agent cooperation score (0.91) throughout the illustrated assessment procedure (Table 4, Figs. 4–9). According to statistical testing, run-to-run noise alone is unlikely to be responsible for these differences (one-way ANOVA $F(5, 24) = 235.55, p < 0.001$; paired $t = 13.96$ against the strongest baseline). Correlation analysis verified that the strongest internal driver of accuracy is reasoning quality ($r = 0.75$), with execution time negatively correlated with both accuracy and workflow efficiency. In addition to a seven-equation mathematical model (η , ACS, Accuracy, L , RU , CS , ADS) that operationalizes workflow quality in quantifiable, explicable terms instead of through opaque prompt engineering, this paper introduced AIWA-Agent, a seven-agent workflow automation framework that unifies Planning, Reasoning, Memory, RAG-based Retrieval, Tool-Calling, Execution, and Validation agents under a continuous-learning control loop. The research gap noted in Section I-G is directly addressed by this formalization and the embedded Human-in-the-Loop and Explainable AI validation layer: while existing frameworks (Autopsy, Crewel, Lang Graph, and Autopen) offer orchestration primitives, they lack a common quantitative model of collaboration and confidence. The findings indicate that structured, graph-constrained agent coordination (like AIWA-Agent and LangGraph) provides latency and reliability advantages over unconstrained conversational multi-agent orchestration (like Autopen) or single-loop autonomous chaining (like Autopsy) for businesses assessing agented automation, especially as the number of active agents increases (Section X-D). Instead of considering human supervision as an external, post-hoc check, the explicit confidence-gated Validation Agent provides organizations with a useful way to balance automation coverage against compliance and audit needs.

Author Contributions

Sant Kumar Maura and Siddhartha Kumar the study's conceptual design, research methodology, investigation, and manuscript preparation. The author also critically reviewed the manuscript and approved the final version for publication.

Funding

The authors did not receive any specific funding for this work.

Conflicts of Interest

The authors declare no conflicts of interest.

References

- [1]. A. Aswan, N. Shaker, N. Parmer, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," arXiv:1706.03762, 2017. Doe: 10.48550/arXiv.1706.03762.

- [2]. J. Wei, X. Wang, D. Schulman's, M. Busman, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems (Neurons)*, vol. 35, pp. 24824-24837, 2022.
- [3]. S. Yao, J. Zhao, D. Yu, N. Du, I. Shoran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023, arXiv:2210.03629. Doe: 10.48550/arXiv.2210.03629.
- [4]. S. Yao, D. Yu, J. Zhao, I. Shoran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," in *Advances in Neural Information Processing Systems (Neurons)*, vol. 36, 2023.
- [5]. T. Schick, J. Dived-Yu, R. Desist, R. Railing, M. Lamely, E. Hambros, L. Zettlemoyer, N. Concede, and T. Siloam, "Tool former: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems (Neurons)*, vol. 36, pp. 68539-68551, 2023.
- [6]. N. Shinn, F. Casino, E. Berman, A. Goliath, K. Narasimhan, and S. Yao, "Reflex ion: Language agents with verbal reinforcement learning," arXiv: 2303.11366, 2023. Doe: 10.48550/arXiv.2303.11366.
- [7]. Y. Sheen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhan, "HuggingGPT: Solving AI tasks with Catgut and its friends in Hugging Face," arXiv:2303.17580, 2023. Doe: 10.48550/arXiv.2303.17580.
- [8]. P. Lewis, E. Perez, A. Pintos, F. Petron, V. Capuchin, N. Goal, H. Kittler, M. Lewis, W. Yam, T. Rocktaschel, S. Riedel, and D. Kiel, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (Neurons)*, vol. 33, pp. 9459-9474, 2020.
- [9]. Y. Ago, Y. Xing, X. Ago, K. Jiao, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," arXiv:2312.10997, 2024. Doe: 10.48550/arXiv.2312.10997.
- [10]. L. Wang, C. Ma, X. Fang, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al., "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, art. 186345, 2024. Doe: 10.1007/s11704-024-40231-1.
- [11]. Z. Xi, W. Chen, X. Goo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, et al., "The rise and potential of large language model based agents: A survey," *Science China Information Sciences*, vol. 68, no. 2, art. 121101, 2025. Doe: 10.1007/s11432-024-4222-0.
- [12]. Q. Wu, G. Banal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "Autopen: Enabling next-gen LLM applications via multi-agent conversations," in *Proc. 1st Conf. Language Modeling (COLM)*, 2024, arXiv:2308.08155. Doe: 10.48550/arXiv.2308.08155.
- [13]. K.-T. Tran, D. Dao, M.-D. Nguyen, Q.-V. Pham, B. O'Sullivan, and H. D. Nguyen, "Multi-agent collaboration mechanisms: A survey of LLMs," arXiv: 2501.06322, 2025. Doe: 10.48550/arXiv.2501.06322.
- [14]. V. Canola, V. Hassidic, A. R. Sultana, D. Ghost, D. Hangar, and B. Skidder, "A review of trustworthy and explainable artificial intelligence (XAI)," *IEEE Access*, vol. 11, pp. 78994-79015, 2023. Doe: 10.1109/ACCESS.2023.3294569.
- [15]. M. Marsha, K. Lam, J. Wood, A. Alchemy, and J. Kaila, "Explainable artificial intelligence: A survey of needs, techniques, applications, and future direction," arXiv: 2409.00265, 2024. Doe: 10.48550/arXiv.2409.00265.
- [16]. E. Cambria, L. Maladroit, F. Mercuric, N. Nubian, and A. Saves, "XAI meets LLMs: A survey of the relation between explainable AI and large language models," arXiv: 2407.15248, 2024. Doe: 10.48550/arXiv.2407.15248.

- [17]. X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning," *Future Generation Computer Systems*, vol. 135, pp. 364-381, 2022. Doe: 10.1016/j.future.2022.05.014.
- [18]. [18] S. Afrin, S. Roxana, and R. Aram, "AI-enhanced robotic process automation: A review of intelligent automation innovations," *IEEE Access*, vol. 12, pp. 186534-186552, 2024. Doe: 10.1109/ACCESS.2024.3513279.
- [19]. J. G. Enriquez, A. Jimenez-Ramirez, F. J. Dominguez-Mayo, and J. A. Garcia-Garcia, "Robotic process automation: A scientific and industrial systematic mapping study," *IEEE Access*, vol. 8, pp. 39113-39129, 2020. Doe: 10.1109/ACCESS.2020.2974934.
- [20]. T. Chakraborti, V. Isahagian, R. Khalid, Y. Khazaeni, V. Muthusamy, Y. Risk, and M. Univac, "From robotic process automation to intelligent process automation: Emerging trends," in *Business Process Management: Block chain and RPA Forum (BPM 2020)*, Springer, 2020, pp. 215-228. Doe: 10.1007/978-3-030-58779-6_15.
- [21]. V. Tope and S. Tube, "AI agented workflows and enterprise APIs: Adapting API architectures for the age of AI agents," *arXiv: 2502.17443*, 2025. Doe: 10.48550/arXiv.2502.17443.
- [22]. D. Finland, F. Fournier, L. Lemonade, I. Skarbovsky, and A. J. E. Swevels, "How well can large language models explain business processes?" *arXiv:2401.12846*, 2024. Doe: 10.48550/arXiv.2401.12846.
- [23]. M. Chui, E. Has an, R. Roberts, A. Single, K. Same, A. Sukharevsky, L. Yee, and R. Zimmer, "The economic potential of generative AI: The next productivity frontier," *McKinsey Global Institute Report*, 2023.
- [24]. K. Gurnard, A. Jangra, H. Baber, and M. M. Islam, "An analytical review on the impact of artificial intelligence on the business industry: Applications, trends, and challenges," *IEEE Engineering Management Review*, vol. 52, no. 2, pp. 45-58, 2024. Doe: 10.1109/EMR.2024.3355973.
- [25]. Open-air, J. Achaia, S. Adler, S. Agawam, et al., "GPT-4 technical report," *arXiv: 2303.08774*, 2023. Doe: 10.48550/arXiv.2303.08774.
- [26]. H. Touring, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babied, N. Bashlykov, et al., "Llama 2: Open foundation and fine-tuned chat models," *arXiv:2307.09288*, 2023. Doe: 10.48550/arXiv.2307.09288.
- [27]. W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hour, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al., "A survey of large language models," *arXiv:2303.18223*, 2023. Doe: 10.48550/arXiv.2303.18223.
- [28]. Z. Jib, N. Lee, R. Frisked, T. Yu, D. Su, Y. CSU, E. Ishii, Y. J. Bang, A. Mayotte, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1-38, 2023. Doe: 10.1145/3571730.
- [29]. Crewel Inc., "Crewel: Framework for orchestrating role-playing, autonomous AI agents," *Gather repository*, <https://github.com/crewAIInc/crewAI>, 2024.
- [30]. Lang Chain Inc., "Lang Graph: Building language agents as graphs," *Gather repository and documentation*, <https://github.com/langchain-ai/langgraph>, 2024.