

CRAFT: A Domain-Agnostic Framework for Reliable Multi-Agent AI Systems

Dhanush Gowda H K

Computer Science and Engineering
East West Institute of Technology
Bengaluru, India
dhanushgowdahk04@gmail.com

Dhanush H M

Computer Science and Engineering
East West Institute of Technology
Bengaluru, India
dhanushhm200518@gmail.com

Jeevan V

Computer Science and Engineering
East West Institute of Technology
Bengaluru, India
jeevanvinodh2705@gmail.com

Davan Gowda K Y

Computer Science and Engineering
East West Institute of Technology
Bengaluru, India
davangowdak@gmail.com

Prof. Shubhadip Karan

Computer Science and Engineering
East West Institute of Technology
Bengaluru, India
shubhadipoutstanding4@gmail.com

Abstract—Large language model based agents can perform complex reasoning and tool use, but reliable multi-agent systems require more than simply assigning multiple agents to a task. They must understand task context, select suitable specialists, combine heterogeneous outputs, and evaluate the resulting answer before delivery. This paper presents CRAFT, a domain-agnostic framework organized around five stages: Context Analyzer, Routing, Agent Specialization, Fusion, and Tempered Evaluation, with memory operating across the pipeline. CRAFT is derived from an earlier AgentHub platform proposal that emphasized reusable agents, orchestration, execution, and agent discovery. The framework converts those platform-level ideas into a reusable reasoning and orchestration pattern that can be instantiated for different domains. A reference implementation was developed for financial analysis using specialized agents, external market-data tools, memory, and graph-based orchestration. Preliminary prototype observations demonstrate end-to-end execution and the ability of the evaluation stage to identify a deliberately incorrect financial calculation. These observations establish feasibility rather than superiority over competing systems. The paper formalizes the architecture, describes its implementation, positions it against recent work on agentic AI and LLM-based multi-agent decision making, and defines an evaluation protocol for future controlled experiments.

Keywords—*Agentic AI, multi-agent systems, large language models, orchestration, routing, memory, evaluation, reliability*

I. INTRODUCTION

The emergence of large language models (LLMs) has shifted AI systems from narrowly scripted automation toward systems capable of language-based reasoning, planning, tool use, and adaptive interaction. Agentic AI extends these capabilities by allowing systems to pursue goals over multiple steps with reduced human intervention. Recent surveys describe autonomy, adaptability, goal orientation, resource management, and safe deployment as central characteristics and open challenges of agentic AI [1].

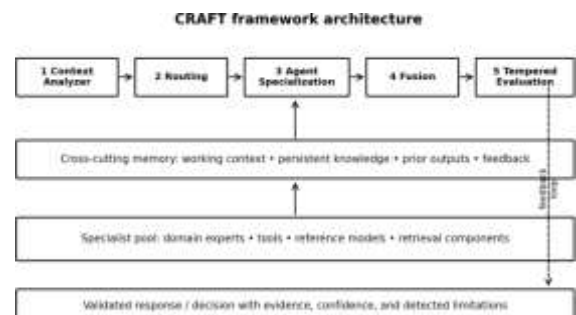
A further challenge appears when multiple LLM-based agents collaborate. Multi-agent decision making introduces communication, coordination, role assignment, and consensus problems that are not present in the same form in single-agent systems [2]. Existing research has investigated role-based agents, communication protocols, autonomous workflows, and dynamic task allocation, while recent autonomous systems surveys emphasize the need for reproducible evaluation, robustness, and secure agent interaction [2]–[4].

Our initial AgentHub proposal addressed this problem from a platform perspective. It envisioned an agent marketplace, orchestration engine, execution environment, task planner, agent registry, and knowledge layer so that agents could be created, discovered, combined, and deployed as reusable digital workers. The proposal also identified agent collaboration, task decomposition, reliability scoring, and resource optimization as research directions.

The present work evolves that platform concept into CRAFT, a reusable framework for constructing domain-specific multi-agent AI applications. Rather than treating orchestration as a collection of ad hoc prompts and agent calls, CRAFT defines a structured pipeline: understand the context, route the task, specialize the work, fuse the results, and evaluate the result before delivery. Memory operates across these stages to preserve relevant state and prior knowledge.

The contributions of this paper are as follows:

- A five-stage, domain-agnostic architecture for structured multi-agent reasoning and orchestration.
- A separation between task understanding, specialist selection, synthesis, and evaluation that supports modular domain instantiation.
- A memory model that operates across the pipeline rather than being confined to an individual agent.
- A reference financial implementation demonstrating specialist agents, external tools, graph-based orchestration, and evaluation.
- A reproducible experimental protocol for measuring routing, answer quality, critic reliability, latency, cost, and human preference.



CRAFT architecture. Memory is cross-cutting across the five processing stages.

II. RELATED WORK

A. Agentic AI

Acharya et al. characterize agentic AI as systems capable of pursuing complex goals with a degree of autonomy and adaptation

in changing environments [1]. Their survey highlights goal alignment, scalability, resource constraints, real-time learning, and governance as important research directions. These concerns motivate a framework that treats autonomy as an engineered pipeline rather than as an unstructured property of an LLM.

B. LLM-Based Multi-Agent Decision Making

Sun et al. distinguish LLM-based multi-agent decision making from conventional single-agent decision making and emphasize the additional difficulty created by inter-agent communication and coordination [2]. Their survey discusses learning-to-coordinate and learning-to-communicate approaches and notes limitations associated with computational demand, hallucination, communication overhead, and scaling. CRAFT addresses these concerns at the application-architecture level by explicitly separating routing, specialization, fusion, and evaluation.

C. Autonomous Workflow Systems

Apuri and Yepuri describe multi-agent autonomous workflow systems with task planning, agent coordination, dynamic delegation, and domain applications [3]. Their architecture emphasizes autonomous workflows and scalability. CRAFT is complementary: it provides a more explicit reasoning lifecycle for what happens inside a reusable multi-agent application after a task enters the system.

D. Autonomous Systems, Evaluation, and Security

Ferrag et al. survey LLM and AI agents in autonomous systems and emphasize applications, datasets, benchmarks, robustness, and security [4]. Their discussion highlights prompt injection, data poisoning, multimodal perturbations, privacy leakage, and vulnerabilities in cooperative agent communication. CRAFT therefore treats evaluation as a first-class stage rather than an optional post-processing step.

III. FROM AGENTHUB TO CRAFT

The relationship between AgentHub and CRAFT is evolutionary rather than interchangeable. AgentHub was initially formulated as a platform for publishing, discovering, orchestrating, and executing reusable agents. Its proposed layers included user interaction, application services, task planning, orchestration, agent registry or marketplace, agent execution, and knowledge and data services. CRAFT abstracts the reasoning lifecycle needed by such an ecosystem into a domain-agnostic pattern.

From the initial AgentHub concept to the CRAFT framework



Evolution from the initial AgentHub platform concept to the CRAFT reasoning pattern.

This distinction is important for positioning the contribution. AgentHub primarily specifies platform infrastructure and an ecosystem model. CRAFT specifies how a complex request can be transformed into a structured multi-agent execution and validation cycle. An AgentHub-like platform can host CRAFT instances, while CRAFT can also be implemented without a marketplace.

IV. CRAFT FRAMEWORK

A. Context Analyzer

The Context Analyzer converts a natural-language request into an explicit problem representation. It identifies the user objective, constraints, domain, required evidence, expected output form, and

relevant context. The stage reduces ambiguity before specialist agents are selected.

B. Routing

The Routing stage maps the interpreted task to one or more suitable specialists. Routing may be rule-based, capability-based, model-assisted, or hybrid. The framework does not require a single routing algorithm; instead, routing is an explicit architectural responsibility whose accuracy can be measured independently.

C. Agent Specialization

Selected specialists perform complementary tasks using domain prompts, tools, reference models, retrieval components, or external APIs. Specialists are intended to be replaceable modules rather than fixed personas. For a financial instantiation, examples include market-data retrieval, fundamental analysis, technical analysis, risk analysis, macroeconomic context, portfolio analysis, and reporting.

D. Fusion

Fusion combines specialist outputs into a coherent response. The fusion component must reconcile disagreement, preserve supporting evidence, and prevent duplicated or contradictory claims. Fusion can be implemented as a dedicated synthesis agent or as a structured aggregation procedure.

E. Tempered Evaluation

Tempered Evaluation is the final reliability stage. Instead of assuming that a generated synthesis is correct, the system evaluates it against explicit criteria such as factual consistency, numerical correctness, evidence coverage, instruction compliance, and domain-specific constraints. The term tempered emphasizes that the critic is a gate for identifying weaknesses, not an assertion that every generated answer becomes correct.

F. Cross-Cutting Memory

Memory provides persistent and working context across the pipeline. It can retain relevant prior interactions, intermediate results, domain knowledge, and evaluation feedback. Because memory can also propagate errors, it must be treated as a controlled information source rather than an unquestioned truth store.

V. REFERENCE IMPLEMENTATION

A reference implementation was developed for financial analysis to test whether the CRAFT pattern could be realized as an executable application. The implementation uses graph-based orchestration and a set of specialized finance agents. External market-data tooling supplies current data to the relevant specialists, while memory supports state across the workflow.

The implementation contains eight finance-oriented specialist roles. The architecture also supports framework injection so that domain-specific agents can share common orchestration behavior without duplicating the complete workflow. The reference implementation therefore serves as a concrete instantiation of the framework rather than as a claim that finance is the only target domain.

A prototype critic test demonstrated the intended evaluation behavior: an incorrect CAGR calculation introduced into a financial response was detected by the evaluation stage, which returned a high overall score together with specific identified issues. This observation is a functional smoke test, not a statistical reliability estimate.

VI. EVALUATION METHODOLOGY

A rigorous evaluation of CRAFT should compare the full framework with controlled ablations. The proposed experimental matrix contains a single-LLM baseline, CRAFT without routing, CRAFT without specialist decomposition, CRAFT without tempered evaluation, and the full CRAFT pipeline.

The primary metrics are:

Factual accuracy and task-level correctness.

Routing accuracy and specialist-selection precision.
Hallucination and unsupported-claim rate.
Critic reliability, including agreement with expert judgments.
End-to-end latency and token or API cost.
Human preference or expert assessment of final responses.

For critic reliability, agreement statistics such as Cohen's kappa or rank correlation can be reported where the evaluation protocol produces categorical or ordinal judgments. Specialist-selection experiments should use a held-out query set covering multiple task types rather than examples selected only to favor CRAFT.

The current prototype observations are preliminary. In small smoke tests, observed execution times increased with the number of specialists, approximately 45 seconds with one specialist, 80 seconds with two, and 120 seconds with four. These values are environment-specific observations and should not be treated as a general scalability law.

VII. SYSTEM MODEL AND WORKFLOW FORMALIZATION

Let a user request be represented by q and let D denote the available domain context, tools, and persistent memory. CRAFT first constructs a contextual representation $c = C(q, D)$, where C is the Context Analyzer. The Routing stage then produces a selected specialist set $A = R(c)$, rather than invoking every available agent. Each specialist a_i receives the relevant portion of c and produces an output $o_i = a_i(c, T_i)$, where T_i represents the tools or knowledge sources available to that specialist.

The Fusion stage constructs a candidate response $y = F(o_1, \dots, o_n, c)$. The final stage applies an evaluation function E to obtain an assessment $e = E(y, c)$. Depending on the deployment policy, the system may return y together with e , revise y , or reject the response and request additional evidence. This formulation makes the interfaces between stages explicit and allows each stage to be evaluated independently.

A practical routing score can be represented generically as $S(a_i, q) = w_c C_i + w_r R_i + w_p P_i - w_k K_i$, where C_i denotes capability-task compatibility, R_i denotes an empirical reliability estimate, P_i denotes expected performance for the task, K_i denotes computational or monetary cost, and the w terms are deployment-specific weights. This expression is a design abstraction rather than a claim that the current prototype learns these weights.

The workflow can execute specialists sequentially, in parallel, or as a dependency graph. Parallel execution can reduce wall-clock time when tasks are independent, whereas sequential execution is useful when one specialist requires another specialist's output. Consequently, the number of agents alone is not a sufficient measure of system complexity; dependency structure, tool latency, model latency, and context size also affect end-to-end behavior.

VIII. IMPLEMENTATION DETAILS

The reference implementation separates domain-independent orchestration from domain-specific agent definitions. A shared framework layer provides workflow state and execution interfaces, while the financial domain supplies specialist prompts, data tools, and analysis instructions. This separation allows an agent to be replaced or extended without rewriting the complete orchestration pipeline.

The financial instantiation contains eight specialist roles covering complementary analytical functions. The workflow can obtain external market information through tools before specialist reasoning is performed. The final synthesis is then passed to the evaluation component. Memory is available to preserve relevant state between stages, but stored information is not treated as automatically correct; retrieved or remembered claims remain subject to evaluation.

The implementation also illustrates why reliability must be measured at multiple levels. An individual specialist may produce a plausible local answer while the combined response can still contain inconsistent claims. Conversely, a critic may identify a numerical problem without being able to correct it. CRAFT therefore separates generation from evaluation and records the evidence needed for later analysis.

IX. EXPERIMENTAL DESIGN

The proposed evaluation uses a held-out benchmark of multi-step queries divided by task type and difficulty. Each query is executed under the same model and tool configuration across the baseline and ablation conditions. The full CRAFT system is compared with a single-agent baseline and progressively reduced variants. The objective is not only to compare final answer quality but also to determine which architectural component contributes to the observed behavior.

For routing, ground-truth specialist assignments can be established by expert annotation for a subset of queries. Precision, recall, and selection accuracy can then be measured. For final-answer quality, expert judges can score correctness, completeness, evidence use, and instruction adherence using a fixed rubric. For evaluation reliability, critic decisions can be compared against expert labels to estimate false-positive and false-negative rates.

Operational measurements should include wall-clock latency, number of model calls, token consumption, tool calls, and failure or retry counts. These measurements are necessary because additional agents may improve coverage while simultaneously increasing computational cost. A useful deployment decision therefore requires a quality-cost analysis rather than a quality-only comparison.

System	Routing	Specialists	Evaluation	Purpose
Single LLM	—	1	—	Baseline
CRAFT-Eval	✓	✓	✗	Ablation
CRAFT-Route	✗	✓	✓	Ablation
Full CRAFT	✓	✓	✓	Complete pipeline

TABLE I. PROPOSED CONTROLLED COMPARISON MATRIX

X. DOMAIN GENERALIZATION

CRAFT is intended as a pattern rather than a finance-specific architecture. A domain instantiation defines its specialist pool, tools, evidence sources, routing criteria, fusion policy, and evaluation rubric while preserving the five-stage lifecycle. Candidate applications include research assistance, software engineering, healthcare decision support, legal document analysis, education, and scientific workflows. These domains are design targets rather than evaluated results in this paper.

The separation is also useful for maintenance. A specialist can be replaced without redesigning the complete pipeline, and the evaluation stage can be adapted to a new domain without changing the underlying routing or memory abstractions.

XI. DISCUSSION

The central design choice in CRAFT is to make orchestration responsibilities explicit. Many multi-agent systems can be described as collections of agents communicating through prompts. CRAFT instead treats the lifecycle around those agents as a first-class object. This creates measurable interfaces: the context stage can be evaluated for problem understanding, routing for selection accuracy, specialists for task performance, fusion for synthesis quality, and evaluation for error detection.

This decomposition does not eliminate the fundamental limitations of LLM-based agents. Additional specialists can increase latency and cost; communication can introduce irrelevant information; memory can preserve incorrect information; and evaluators can themselves make mistakes. These limitations are consistent with concerns reported in the multi-agent decision-making and autonomous-systems literature [2], [4].

XII. LIMITATIONS AND THREATS TO VALIDITY

First, the present evidence is based on a reference implementation and preliminary prototype observations rather than a large benchmark. Second, the finance implementation does not establish that CRAFT is superior to other multi-agent frameworks. Third, the choice of specialists, prompts, models, tools, and evaluation criteria can strongly influence results. Fourth, latency measurements are sensitive to model provider, network conditions, parallelism, caching, and tool response time.

A further limitation is evaluator dependence. A critic is not an independent oracle simply because it is separated from the generating agent. Future experiments should therefore include human or expert references and should report critic false positives and false negatives rather than only critic scores.

XIII. CONCLUSION AND FUTURE WORK

This paper presented CRAFT, a domain-agnostic framework for structuring reliable multi-agent AI applications. CRAFT organizes execution into Context Analyzer, Routing, Agent Specialization, Fusion, and Tempered Evaluation, with memory operating across the pipeline. The framework emerged from the earlier AgentHub platform proposal by refining its platform-level concepts into an explicit reusable reasoning lifecycle.

The reference financial implementation demonstrates that the architecture can be executed with specialized agents, external data tools, memory, and graph-based orchestration, while a prototype evaluation test demonstrates the ability to identify a deliberately incorrect financial calculation. The next stage of the work is empirical: controlled baseline comparisons, routing evaluation, ablation studies, critic reliability measurement, multi-domain benchmarks, cost analysis, and security testing are required to establish the practical value and limitations of CRAFT.

XIV. ACKNOWLEDGMENT

The authors used generative AI assistance during manuscript drafting, organization, and language editing. The authors are responsible for verifying the technical content, references, experimental claims, and final manuscript. Any IEEE submission should retain an AI-use disclosure consistent with the target venue's current policy.

REFERENCES

- D. B. Acharya, K. Kuppan, and B. Divya, —Agentic AI: Autonomous intelligence for complex goals—A comprehensive survey,| IEEE Access, vol. 13, 2025, doi: 10.1109/ACCESS.2025.3532853.
- C. Sun, S. Huang, and D. Pompili, —LLM-based multi-agent decision-making: Challenges and future directions,| IEEE Robotics and Automation Letters, vol. 10, no. 6, pp. 5681–5688, 2025, doi: 10.1109/LRA.2025.3562371.
- H. Apuri and C. Yepuri, —Design of multi agent autonomous workflow systems using agentic AI frameworks,| in Proc. Int. Conf. on AI-Driven Smart Systems and Ubiquitous Computing (ICAUC), 2026, pp. 1731–1737, doi: 10.1109/ICAUC68182.2026.11441280.
- M. A. Ferrag, A. Lakas, N. Tihanyi, and M. Debbah, —LLM and AI agents for autonomous systems: A survey of applications, datasets, and security challenges,| IEEE Open Journal of the Intelligent Transportation Systems, vol. 7, 2026, doi: 10.1109/OJITS.2026.3665677.