International Journal of Computer Techniques – Volume 12 Issue 6, August - 2025

Human-in-the-Loop (HITL) Orchestration for Agentic Use-Cases

A Practical Framework for Supervising Autonomous AI Agents in Production Environments

Sandeep Reddy Kaidhapuram
Independent Researcher, Austin, TX
sandeep.kaidha@gmail.com

Abstract: **- The rapid maturation of large language model agents during the past two years has forced enterprises to reckon with a difficult question: how much autonomy should these systems actually have before a human checks the work? Agentic systems now plan multi-step tasks, call tools, write code, and in some cases transact on behalf of users. When those actions are reversible and low-stakes, full automation is fine. When they are not, somebody needs to be in the room. This paper examines human-in-the-loop (HITL) orchestration as a design pattern for governing agentic use cases, drawing on deployment patterns observed across software engineering, customer operations, finance, and healthcare between 2023 and late 2025. We propose a tiered orchestration model that treats human intervention not as an exception but as a first-class component of the agent runtime, with explicit checkpoints, confidence-driven escalation, and reviewer feedback channels. We evaluate the model through a literature review of published research and industry reports, compare it with leading orchestration frameworks, and present analysis of practical deployments. Results suggest that well-designed HITL layers improve task accuracy, reduce liability exposure, and, somewhat counterintuitively, increase overall throughput when measured across end-to-end business outcomes rather than raw model calls.**

Keywords: **- Human-in-the-loop, HITL, agentic AI, LLM orchestration, autonomous agents, AI governance, oversight systems, workflow automation, approval workflows, confidence calibration, reinforcement learning from human feedback, agent safety.**

I. INTRODUCTION

A year ago, I watched a development team at a mid-sized insurance firm demo an agent they had built for claims triage. It was genuinely impressive. The agent could read a policy, pull historical claims, cross-reference medical records, and produce a recommendation for approval or denial in under forty seconds. Then the senior underwriter in the room asked a simple question: who signs off before the money goes out? Nobody had a clean answer. The project stalled for three months while the team retrofitted approval gates, audit trails, and a reviewer interface. That story is not unusual. It is, in fact, the defining operational problem for agentic AI right now.

Agentic systems differ from earlier generations of AI in a structural way. Classifiers predict. Chatbots respond. Agents act. They decompose goals, invoke tools, persist state, and chain outputs across many steps. Each of those steps is an opportunity for something to go wrong, and because agents compound small errors into large ones, unsupervised execution at scale is an unusually risky design choice. The field has converged on a name for the corrective pattern: human-in-the-loop orchestration, or HITL for short.

The term itself is older than the current agent wave. HITL machine learning has been around since the early 2010s, usually in the context of active learning, where a human labels the examples the model is least sure about. What is new is that HITL has moved from training time to runtime. We are no longer just asking humans to help a model learn. We are asking them to supervise a system that is taking real actions in the world and doing so in a way that does not strangle the efficiency gains that motivated automation in the first place.

This paper argues that HITL orchestration, when treated as a deliberate architectural layer rather than a safety bolt-on, is one of the most important design problems in applied AI today. The rest of the paper is organized as follows. Section II reviews existing research and industry practice around HITL and agentic workflows. Section III proposes a tiered orchestration model that distinguishes between pre-execution approval, in-execution intervention, and post-execution audit. Section IV presents comparative analysis across deployment patterns we have observed or that have been published by vendors and research teams. Section V concludes with lessons that seem, at least so far, to generalize.

A note on scope. This paper focuses on agents built around large language models, since that is where the operational challenges are most acute right now. The patterns discussed here also apply to robotic process automation, symbolic planners, and hybrid systems, but the specific failure modes differ. Where the distinctions matter, we call them out.

It is worth pausing on the word "orchestration" because it is doing a lot of work in the title of this paper. We do not mean merely the mechanical sequencing of agent steps. We mean the broader set of design decisions that govern when the agent acts, when it asks, when it stops, and who gets told. A good orchestration layer is to an agent what a conductor is to an orchestra. The musicians know how to play. The conductor decides when they play, how loud, and when to rest. Remove the conductor, and a handful of soloists will produce something interesting. Thirty musicians will produce noise. Agent systems past a certain complexity behave the same way, and this is the gap HITL orchestration is trying to close.

II. LITERATURE REVIEW

The literature on HITL spans at least three distinct traditions that do not always speak to each other. The oldest is active learning, which treats human labels as a scarce resource to be queried economically. The second is interactive machine learning, which is concerned with how humans and models collaborate in real time. The third, and most recent, grew out of the RLHF revolution and focuses on using human preferences to shape the model itself. Agentic deployments draw on all three. In what follows we trace the most relevant threads and try to show where they connect.

A. The Origins of Human-in-the-Loop Machine Learning

Early HITL literature is dominated by active learning, where the goal is to minimize labeling cost. Work by Settles and others in the late 2000s established the now-standard idea that query strategies, such as uncertainty sampling or query-by-committee, let a model ask for labels on the examples that will improve it the most. The implicit assumption was that humans were expensive and slow, so the model should be frugal with their attention. That assumption still holds, but it has been reframed. In agentic systems, the human is not expensive because they cost money to hire. They are expensive because every moment they spend reviewing an agent step is a moment the agent is idle.

B. Interactive Machine Learning and Mixed-Initiative Systems

A parallel research tradition, sometimes called mixed-initiative interaction, goes back further. Horvitz's work at Microsoft Research in the late 1990s laid out principles for when a system should take initiative and when it should defer. Many of those principles translate directly to modern agent design. In particular, the idea that a system should estimate the cost of being wrong and the cost of interrupting the user, then act accordingly, is the core of what we now call confidence-based escalation. The vocabulary has changed. The math has not.

C. RLHF, DPO, and Learning from Preferences

The modern wave of large language models is inseparable from reinforcement learning from human feedback, or RLHF. Christiano and colleagues in 2017 demonstrated that preference-based learning could align agents with human intent in ways that scalar reward functions could not. Later work on direct preference optimization and on constitutional AI extended the toolkit. For the purposes of this paper, the key point is that these techniques shape the agent before deployment. They do not replace runtime oversight. An agent trained with RLHF may be more polite and less prone to obvious mistakes, but it will still take bad actions when given ambiguous instructions, especially when it is chaining tool calls across a long horizon.

D. Agent Frameworks and Orchestration Layers

Between 2023 and 2025, an ecosystem of agent frameworks emerged. LangGraph, CrewAI, AutoGen, LangChain's own agent primitives, and Anthropic's tool-use APIs all offered ways to compose agents from models, tools, memory stores, and routing logic. Most of them treat human intervention as a node in a graph: an interrupt that can be inserted between model calls. This is progress, but the abstraction is usually weak. The framework gives you a way to pause; it typically does not give you a way to reason about when to pause, who should see the pause, what context they need, or how to feed their response back into the agent's working memory without corrupting it.

More recent work, including Anthropic's research on computer-use agents and Google DeepMind's publications on agentic evaluation, has begun treating oversight as a first-class concern. The practitioner literature, including posts by teams at Shopify, Stripe, and several large banks, has converged on a similar set of patterns. We synthesize these below and in the proposed model.

E. Governance, Regulation, and Audit

The regulatory environment caught up quickly. The EU AI Act, finalized in 2024, classified several agentic use cases, particularly in employment and credit, as high-risk and required documented human oversight. The NIST AI Risk Management Framework in the United States pushed in a

similar direction. In financial services, regulators began asking explicitly how model outputs were reviewed before affecting customers. The research literature here is less technical and more procedural, but it matters: if your HITL design cannot produce an audit trail a regulator will accept, it does not really exist. We have seen otherwise solid agent architectures fail their first compliance review because the reviewer interface produced no durable record of who saw what and when.

F. Evaluation Frameworks for Agentic Systems

Evaluation of agentic systems is a young field, and the published benchmarks reflect that. Work on WebArena, AgentBench, and SWE-bench during 2023 and 2024 gave the community a way to compare agent implementations on constrained tasks. These benchmarks are useful for model selection and for detecting regressions, but they are poor proxies for the operational question a business actually cares about. A benchmark that measures whether an agent can close a GitHub issue does not tell you what happens when that agent closes the wrong issue at two in the morning because a malformed webhook triggered a retry. The evaluation literature has begun to acknowledge this gap. Recent papers on agentic evaluation propose longer-horizon tasks, inject adversarial conditions, and measure graceful degradation rather than only peak accuracy. The trend is welcome, but the field has not yet converged on a standard way to measure oversight quality, which is precisely what HITL orchestration aims to provide.

G. Gaps in the Literature

Two gaps stand out. First, most published work treats HITL as either a training-time mechanism or a coarse runtime gate. There is relatively little on the orchestration logic that decides when to invoke a human, what to show them, and how to merge their decision back into an ongoing multi-step task. Second, evaluation metrics for HITL systems are immature. Accuracy on benchmark tasks does not capture the operational question: across an end-to-end process, does adding the human actually improve outcomes, and at what cost in latency and reviewer fatigue? This paper attempts to address both gaps.

III. PROPOSED MODEL: A TIERED HITL ORCHESTRATION FRAMEWORK

The framework below is the product of designing, reviewing, or debugging somewhere between fifteen and twenty agent systems across the past two years. It is not a formal theorem. It is a set of structural commitments that have held up across domains. The core idea is straightforward. HITL should not be a single decision. It should be a set of decisions distributed across three phases of agent execution, each with different cost profiles, different reviewer audiences, and different technical primitives.

A. The Three Phases

We divide the agent lifecycle into three phases: pre-execution, in-execution, and post-execution. Each phase has a characteristic form of human oversight.

Pre-execution oversight happens before the agent takes any irreversible action. It looks like approval workflows, plan review, and scope confirmation. The classic example is a coding agent that proposes a plan and waits for the developer to say yes before it starts writing files. Pre-execution oversight is the cheapest to implement because it fits neatly into existing review habits, and it is the easiest to sell to risk committees because it feels familiar. The cost is latency: the agent cannot proceed until the human responds.

In-execution oversight happens mid-task, typically when the agent encounters an action it flags as risky or ambiguous. It looks like tool-call approval, escalation to a supervising agent, or a pop-up that asks the human to confirm a specific call. This is the most technically interesting phase because it requires the agent to reason about its own uncertainty, which remains an unsolved problem. In practice, teams fall back on heuristics: certain tool names always escalate, certain confidence thresholds always escalate, and certain monetary limits always escalate.

Post-execution oversight is an audit. The agent completes its task; a human reviews a sample of outputs after the fact. This is cheap at the level of individual tasks but essential for building the longitudinal data that lets you tune the first two phases. If you do not have good post-execution review, you cannot calibrate your in-execution thresholds, and your system will drift.

B. Confidence-Driven Routing

The routing logic that decides which phase applies to a given action is the heart of the orchestration layer. In the simplest case, it is a static table: actions of type A always require pre-execution approval; actions of type B require in-execution confirmation above a dollar threshold; actions of type C are audited weekly. This works, and it is where most teams should start. Over time, a more adaptive layer emerges, in which the agent's reported confidence, the criticality of the action, and the reviewer's historical approval rate all feed into the routing decision.

One practical note. LLM-reported confidence is noisy and often mis calibrated. Several research groups have

documented that models are systematically overconfident on tasks they are poor at and underconfident on tasks they handle well. Do not use raw token probabilities as your confidence signal. Use ensembles, self-consistency checks, or an external verifier model. The engineering cost is real, but the accuracy gain is substantial.

C. The Reviewer Interface

The reviewer interface is where good HITL designs live or die. We have seen brilliant orchestration layers fail because the reviewer was handed a wall of JSON and asked to approve a database mutation at 4:50 PM on a Friday. The interface should surface three things and nothing else in its default view: what the agent is about to do, why it thinks that action is correct, and what reversing the action would cost. Deeper context should be available on demand, but the top of the screen should let a human make a decision in under ten seconds for routine cases.

A useful rule of thumb is that the review cost of a single item should be less than ten percent of the time saved by automating the preceding steps. If your agent saves twenty minutes and the review takes three, you are losing the economic argument.

D. Feedback Ingestion

Reviewer decisions are some of the highest-quality training data you will ever collect. A human approving or rejecting an agent action, with a brief reason attached, is almost the definition of a preference label. Your orchestration layer should write these decisions to a structured store and feed them back into evaluations, fine-tuning pipelines, and prompt revisions. Without this loop, the system cannot improve.

A caveat. Not all reviewer decisions are good data. Fatigue, rubber-stamping, and divergent reviewer standards are all real. Build in cross-review for a sample of items, track per-reviewer agreement rates, and retire reviewers from the feedback loop when their agreement with consensus falls below a threshold. This sounds harsh, but it is the same quality control that any human annotation pipeline needs.

E. The Five-Tier Action Taxonomy

To operationalize routing, we recommend classifying every possible agent action into one of five tiers based on reversibility and blast radius. The tier determines which oversight phase applies.

TABLE I
FIVE-TIER AGENT ACTION TAXONOMY

Tier	Action Class	Example	Oversight
T0	Read-only	Searching a knowledge base	None; audit sample only
T1	Reversible internal	Drafting a document, filing a ticket	Post-execution audit
T2	Reversible external	Sending an internal email, creating a calendar event	In-execution confirmation above threshold
T3	Hard to reverse	Customer messaging, code merges, data mutations	Pre-execution approval required
T4	Irreversible or regulated	Payments, contract execution, medical recommendations	Pre-execution approval plus secondary reviewer

The classification is deliberately coarse. Fine-grained taxonomies look elegant in design documents but break down in practice because engineers and reviewers cannot remember them. Five tiers is a workable compromise between expressiveness and cognitive load.

F. A Worked Example: The Refund Agent

To make the framework concrete, consider a refund-processing agent for an e-commerce platform. The agent reads an incoming customer complaint, consults the order history, evaluates return policy eligibility, and either issues a refund, offers a replacement, or escalates to a human. Each of these actions can be placed in the five-tier taxonomy. Reading the order history is a T0 read. Drafting the reply to the customer is a T1 reversible internal action. Sending the reply is a T2 reversible external action, because a sent email can be followed up but cannot be unsent. Issuing a refund under one hundred dollars might be a T2 action, while refunds above that threshold become T3. Refunds above a regulatory or accounting threshold, or refunds that touch flagged accounts, become T4 and require a secondary reviewer. A well-orchestrated version of this agent has a single policy table that maps every combination of amount, account state, and customer tier to an oversight phase, and the table is owned by the business, not the engineers.

In practice, when this agent was first rolled out at a retailer we worked with, the team made a classic mistake. Every outbound message was routed to in-execution confirmation. Within two weeks the reviewer queue had backed up to several hundred items on quiet days and several thousand on busy ones. The fix was not to loosen oversight globally. The fix was to split the taxonomy. Messages that were purely informational, such as shipping status updates generated from structured data, moved to T1 and became audit-only. Messages that contained monetary offers remained at T2. Messages that apologized for a service failure, which

carried reputational risk, were pushed to T3. The queue halved within a week. This kind of surgical adjustment is what the tiered framework is designed to make possible.

G. Failure Modes the Framework Guards Against

Three failure modes are common enough to warrant explicit mention. The first is silent drift, where the agent begins producing slightly worse outputs over time as its inputs shift and nobody notices because the reviewer sample rate is too low. The second is escalation collapse, where so many items require review that the reviewer queue grows unboundedly and the human becomes a bottleneck. The third is trust inversion, where reviewers start approving agent actions without reading them because the agent has been right a hundred times in a row. Each tier of the framework attempts to raise the cost of one of these failures: audit sampling catches drift, confidence-based routing controls queue depth, and secondary review on T4 actions breaks the pattern of rubber-stamping.

IV. RESULTS AND ANALYSIS

A. Comparative Survey of Deployment Patterns

To test the framework against reality, we surveyed twelve deployed agentic systems for which public documentation or first-hand access was available. The domains included software engineering (four systems), customer support (three), internal IT operations (two), financial back-office (two), and clinical documentation (one). The goal was not statistical significance; the sample is far too small for that. The goal was to see whether the tiered framework captured the distinctions that mattered in practice.

The short answer is yes, with caveats. Every system we surveyed had at least one oversight layer. The systems that reported the best operational outcomes, measured by reviewer approval rate trajectory and by net hours saved per week, had layers at all three phases. Systems with only pre-execution approval tended to be perceived as slow; systems with only post-execution audit tended to accumulate incidents; systems with only in-execution confirmation tended to suffer from alert fatigue because the confidence signal was poorly calibrated.

TABLE II
SURVEY OF DEPLOYED HITL SYSTEMS

Domain	Systems Surveyed	Dominant Tier	Observed HITL Cost
Software engineering	4	T3	Low, merged with code review
Customer support	3	T2–T3	Moderate, scales with volume
Internal IT operations	2	T3	Low, automation-heavy
Financial back-office	2	T3–T4	High, regulatory-driven
Clinical documentation	1	T4	Very high, mandatory sign-off

B. Latency and Throughput Effects

A common objection to HITL is that it destroys the throughput gains that motivated the agent in the first place. The data we collected suggests this fear is overstated when the framework is applied thoughtfully. In the software engineering deployments, for example, adding pre-execution plan review increased median task latency by about 35 percent but reduced post-merge defect rates by roughly 60 percent. The net effect on delivered-value-per-hour was positive, not negative, once downstream bug-fixing time was included in the measurement.

The customer support deployments were more mixed. Adding in-execution confirmation on outbound messages to customers improved tone accuracy and reduced escalations to supervisors, but it increased reviewer workload enough that one team had to hire additional headcount. In that case the HITL layer paid for itself through avoided customer churn, but the payback was measurable only over several quarters, not immediately.

The clinical documentation case was the most interesting. The agent drafted visit notes for physician review. The HITL layer was mandatory under healthcare regulations, so the question was not whether to have it but how to structure it. Moving from a dense full-note review to a highlighted-diff view, where the agent flagged its own lowest-confidence sections, cut physician review time per note from four minutes to about ninety seconds. The underlying agent did not change. Only the reviewer interface changed. That is the point.

C. Calibration and Trust Over Time

Across deployments we observed a consistent pattern in how reviewer trust evolved. In the first two to four weeks, reviewers were cautious and approval rates were low. In weeks four through twelve, as the agent demonstrated competence on routine cases, approval rates rose sharply and review times dropped. Then, somewhere between weeks twelve and twenty, a second inflection appeared. Either the team noticed that reviewers were no longer catching real errors, or they noticed that reviewers had started catching

more errors as the agent encountered edge cases outside its training distribution. The first pattern signals trust inversion and demands intervention. The second pattern is healthier but still requires the team to revisit training data and routing logic.

The practical lesson is that HITL is not a steady state. It is a moving relationship between a reviewer and an agent, and the orchestration layer has to expose enough telemetry for the team to see the relationship shifting. We recommend tracking three metrics weekly at a minimum: per-tier approval rate, per-tier median review time, and disagreement rate on the cross-reviewed sample. A sharp change in any of these signals a problem that cannot be ignored.

D. Regulatory Acceptance

The two financial back-office systems in the survey both passed internal compliance review on their first submission. In both cases, the teams had explicitly designed the orchestration layer to produce audit artifacts: signed reviewer decisions with timestamps, immutable logs of agent actions, and clear lineage from policy rule to agent prompt to reviewer prompt. This is more engineering work than most teams expect. It is also the difference between a system that ships and one that stalls. The clinical documentation system required an additional HIPAA review but was structurally similar.

E. Cost Structure and Economic Trade-offs

Teams repeatedly underestimate the total cost of a HITL layer. The headline cost is reviewer time, and that is usually budgeted for. The costs that surprise teams are the orchestration engineering, the interface iteration, the telemetry pipeline, and the ongoing calibration work. Across the twelve systems surveyed, reviewer salaries accounted for roughly 40 to 55 percent of the recurring cost of the HITL layer. The remaining share was split between infrastructure, including logging, queueing, and audit storage, and engineering time spent tuning the routing rules. A useful budgeting heuristic is that if you spend one dollar on the agent, you should expect to spend somewhere between 30 and 80 cents on the HITL layer during the first year, dropping to 15 to 40 cents by the second year as the calibration stabilizes. Teams that budget less tend to cut corners on telemetry, which then makes later calibration much harder.

There is also an opportunity cost worth naming. Every hour a senior reviewer spends approving routine agent outputs is an hour they are not spending on work that is genuinely difficult. One of the promising patterns we saw was using junior reviewers for T1 and T2 decisions and reserving senior reviewers for T3 and T4, with a clear escalation path between the two. This matches how human-only workflows already distribute work, but it is easy to forget when the system is being designed around the agent rather than around the people it is meant to support.

F. A Note on Reviewer Experience

Something that surprised us across the survey was how much reviewer satisfaction mattered for long-term system health. Reviewers who felt the agent was a tool they controlled stayed engaged and caught real errors. Reviewers who felt the agent was pushing work at them disengaged, rubber-stamped, and missed errors. The design decisions that separated these two groups were small but consistent. Letting the reviewer edit the agent's output inline, rather than just approve or reject, made a large difference. Showing the reviewer why they were being asked, rather than simply asking, made a similar difference. If you treat reviewers as labelers, they will behave like labelers. If you treat them as collaborators, they will behave like collaborators.

V. CONCLUSION

Agentic AI is not going to reach its potential if we keep treating human oversight as a nuisance we tolerate until the models get better. The models are getting better, faster than we expected a year ago, but they are not approaching the point where autonomous execution on high-stakes tasks is wise. More importantly, even when they do approach that point, the organizations deploying them will still need defensible audit trails, change-control processes, and ways to answer to regulators, customers, and their own boards. HITL orchestration is the architectural answer to all of those needs, and it deserves to be designed with the same care we bring to the models themselves.

The tiered framework proposed here is not the final word. It is a starting structure. Three things seem robust across the deployments we studied. First, HITL should be distributed across pre-execution, in-execution, and post-execution phases, rather than concentrated in any one of them. Second, routing decisions should combine a coarse action taxonomy with an adaptive confidence signal, and teams should be honest about the limitations of LLM-reported confidence. Third, the reviewer interface is as important as the orchestration logic behind it, and treating reviewers as collaborators rather than rubber stamps pays compound returns over time.

Several research directions remain open. Calibration of agent confidence under long-horizon tool use is one. Reliable automated detection of reviewer fatigue and rubber-stamping is another. Cross-agent oversight, where one agent audits another and escalates only exceptions to a human, is a third, and it is where we expect the most interesting work to emerge over the next year. Each of these would reduce the human

load without sacrificing the oversight guarantees that make HITL worth building in the first place.

For the practitioner reading this with a project in flight, the advice is practical. Start with a five-tier taxonomy and assign every action the agent can take to a tier. Build the reviewer interface before the orchestration logic, because the interface will surface the right questions about what the orchestration actually has to do. Instrument everything, because you cannot tune what you cannot measure. And remember that the goal is not to keep humans in the loop forever. The goal is to keep humans in the loop long enough to earn the right to take them out, one tier at a time.

There is a broader cultural point that is harder to capture in a framework. Teams that have been successful with agentic deployments share a certain temperament. They are genuinely curious about what the agent is getting wrong, rather than defensive about it. They treat reviewer complaints as data rather than noise. They are willing to slow down when the metrics say slow down, and they are willing to expand the agent's scope when the metrics say expand. This disposition is not something you can buy or architect. But it is something a leadership team can model, and in every organization where we have seen agentic systems thrive, someone senior was doing exactly that.

Ultimately, HITL orchestration is a wager on the proposition that human judgment and machine scale are complementary rather than competitive. The wager is not new. It is the same one that every generation of automation has made, from the assembly line to the spreadsheet to the compiler. What is new is the speed at which agents can now produce work that looks right and is not—and the corresponding need for orchestration layers that make oversight efficient rather than sporadic. Built well, these layers are what will let organizations deploy agents with confidence. Built poorly, or skipped entirely, they will be the reason the current wave of agentic AI produces more incidents than value. The choice is an engineering one, but it is also a choice about how much care an organization wants to put into its most consequential automated decisions.

CONFLICTS OF INTEREST

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

FUNDING STATEMENT

No external funding was received for the research and publication of this article.

ACKNOWLEDGMENT

The author thanks the reviewers and peers whose feedback improved earlier drafts of this manuscript.

REFERENCES

[1] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the People: The Role of Humans in Interactive Machine Learning," *AI Magazine*, vol. 35, no. 4, pp. 105–120, 2014.

[2] Y. Bai, A. Jones, K. Ndousse, et al., "Constitutional AI: Harmlessness from AI Feedback," *arXiv preprint arXiv:2212.08073*, 2022.

[3] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep Reinforcement Learning from Human Preferences," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[4] European Parliament, "Regulation (EU) 2024/1689 on Artificial Intelligence (Artificial Intelligence Act)," *Official Journal of the European Union*, 2024.

[5] E. Horvitz, "Principles of Mixed-Initiative User Interfaces," in *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, 1999, pp. 159–166.

[6] D. Kaur, S. Uslu, K. J. Rittichier, and A. Durresi, "Trustworthy Artificial Intelligence: A Review," *ACM Computing Surveys*, vol. 55, no. 2, pp. 1–38, 2022.

[7] P. Lewis, E. Perez, A. Piktus, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[8] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, 2023.

[9] L. Ouyang, J. Wu, X. Jiang, et al., "Training Language Models to Follow Instructions with Human Feedback," in *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[10] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct Preference Optimization: Your Language Model Is Secretly a Reward Model," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[11] B. Settles, "Active Learning Literature Survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2010.

[12] B. Shneiderman, "Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy," *International Journal of Human–Computer Interaction*, vol. 36, no. 6, pp. 495–504, 2020.

[13] T. Wu, M. Terry, and C. J. Cai, "AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts," in *Proc. CHI Conf. on Human Factors in Computing Systems*, 2022.

[14] Z. Xi, W. Chen, X. Guo, et al., "The Rise and Potential of Large Language Model-Based Agents: A Survey," *arXiv preprint arXiv:2309.07864*, 2023.

[15] K. Yang, Y. Liu, A. Chaudhary, R. Fakoor, P. Chaudhari, G. Karypis, and H. Rangwala, "Orchestrating Agents: A Survey of Multi-Agent LLM Systems," *arXiv preprint arXiv:2402.01680*, 2024.

[16] D. Zhou, N. Schärli, L. Hou, et al., "Least-to-Most Prompting Enables Complex Reasoning in Large Language Models," in *Proc. International Conference on Learning Representations*, 2023.