# STRIDE: A Systematic Framework for Selecting AI Modalities—Agentic AI, AI Assistants, or LLM Calls

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

The rapid shift from stateless large language models (LLMs) to autonomous, goal-driven agents raises a central question: *When is agentic AI truly necessary?* While agents enable multi-step reasoning, persistent memory, and tool orchestration, deploying them indiscriminately leads to higher cost, complexity, and risk.

We present **STRIDE** (Systematic Task Reasoning Intelligence Deployment Evaluator), a framework that provides principled recommendations for selecting between three modalities: (i) direct LLM calls, (ii) guided AI assistants, and (iii) fully autonomous agentic AI. STRIDE integrates structured task decomposition, dynamism attribution, and self-reflection requirement analysis to produce an *Agentic Suitability Score*, ensuring that full agentic autonomy is reserved for tasks with inherent dynamism or evolving context.

Evaluated across *30 real-world tasks* spanning SRE, compliance, and enterprise automation, STRIDE achieved *92% accuracy* in modality selection, reduced unnecessary agent deployments by *45%*, and cut resource costs by *37%*. Expert validation over six months in SRE and compliance domains confirmed its practical utility, with domain specialists agreeing that STRIDE effectively distinguishes between tasks requiring simple LLM calls, guided assistants, or full agentic autonomy. This work reframes agent adoption as a *necessity-driven* design decision, ensuring autonomy is applied only when its benefits justify the costs.

## 1 Introduction

Recent advances have transformed AI from simple stateless LLM calls to sophisticated autonomous agents, enabling richer reasoning, tool use, and adaptive workflows. While this progression unlocks significant value in domains such as site reliability engineering (SRE), compliance, and automation, it also introduces substantial trade-offs in cost, complexity, and risk. A central design challenge emerges: *when agents are truly necessary*, and when are simpler alternatives sufficient?

We distinguish three modalities: (i) **LLM calls**, providing single-turn inference without memory or tools, which is ideal for straightforward query-response scenarios; (ii) **AI assistants**, which handle guided multi-step workflows with short-term context and limited tool access that is suitable for structured processes requiring human oversight; and (iii) **Agentic AI**, which autonomously decomposes tasks, orchestrates tools, and adapts with minimal oversight, which is necessary for complex, dynamic environments requiring independent decision-making. Table 1 contrasts these modalities.

Current practice often overuses agentic AI, deploying autonomous systems even when simpler modalities would suffice. This tendency leads to unnecessary cost, complexity, and risk, particularly in enterprise contexts where reliability and governance are critical. *A principled framework for*

Submitted to Scaling Environments for Agents (SEA) Workshop @ NeurIPS 2025

Table 1: Comparison of AI Modalities

| Attribute | LLM Call | AI Assistant | Agentic AI |
|---|---|---|---|
| Reasoning Depth | Shallow | Medium | Deep |
| Tool Needs | Single | Single/Multiple | Multiple |
| State Needs | None | Ephemeral | Persistent |
| Risk Profile | Low | Medium | High |
| Use Case Example | Exchange rate lookup | Summarize meeting notes | Plan 5-day travel itinerary |

*deciding when agents are truly necessary has been missing,* leaving design-time choices largely intuition-driven rather than evidence-based. While agentic AI unlocks transformative value in domains like SRE, compliance verification, and complex automation, deploying it indiscriminately carries risks:

- **Overengineering:** using agents for simple queries wastes compute and developer effort.
- **Security & compliance risks:** uncontrolled tool use and API calls may leak sensitive data.
- **System instability:** recursive loops and unbounded workflows degrade reliability.

We propose **STRIDE**, a novel framework for *necessity assessment at design time*: systematically deciding whether a given task should be solved with an LLM call, an AI assistant, or agentic AI. STRIDE analyzes task descriptions across four integrated analytical dimensions:

- **Structured Task Decomposition:** Tasks are decomposed into a directed acyclic graph (DAG) of subtasks, systematically breaking down objectives to reveal inherent complexity, interdependencies, and sequential reasoning requirements that distinguish simple queries from multi-step challenges.
- **Dynamic Reasoning and Tool-Interaction Scoring:** STRIDE quantifies reasoning depth together with tool dependencies, external data access, and API requirements, identifying when sophisticated orchestration beyond basic language processing is necessary.
- **Dynamism Attribution Analysis:** Using a *True Dynamism Score (TDS)*, the framework attributes variability to models, tools, or workflow sources, clarifying when persistent memory and adaptive decision-making are required.
- **Self-Reflection Requirement Assessment:** Assesses need for error recovery and meta-cognition, and integrates all factors into an *Agentic Suitability Score (ASS)* that guides the choice of LLM call, assistant, or agent.

This unified methodology ensures that AI solution selection is not an ad-hoc judgment call, but a structured, repeatable process that balances capability requirements with efficiency, cost, and risk management. Just as scaling laws have guided model development by quantifying performance as a function of parameters and data, we argue that analogous principles are needed for *environmental and task scaling*. Not every task requires autonomy: simple queries map to LLM calls, structured processes to guided assistants, and only dynamic, evolving workflows demand full agentic AI. STRIDE introduces such a structured scaling perspective for modality selection.

**Strategic Integration and Impact:** STRIDE acts as a *"shift-left"* decision tool— i.e., it moves critical choices from deployment time to the design phase—embedding modality selection into early workflows. This prevents over-engineering, avoids under-provisioning, and provides defensible criteria for balancing capability, efficiency, computational cost, and risk.

- We introduce **STRIDE**, the first design-time framework for AI modality selection, shifting decisions left in the pipeline.
- We define a novel quantitative **Agentic Suitability Score** with dynamism attribution, balancing autonomy benefits against cost and risk.
- We evaluate STRIDE on 30 real-world tasks across SRE Jha et al. [2025], compliance, and enterprise automation, demonstrating reduced agentic over-deployment by **45%** while improving expert alignment by **27%**.

Beyond efficiency, this framing directly supports responsible AI deployment. By preventing over-engineering, STRIDE reduces unnecessary surface area for errors, governance failures, and hidden costs, while ensuring that truly complex tasks receive the level of autonomy they demand.

## 2 Related Work

Recent advances have expanded AI from simple LLM calls to guided assistants and adaptive agentic systems. While assistants follow structured workflows, agents plan and make inference-time decisions in dynamic environments. This shift has driven research into task complexity, reasoning depth, and self-reflection, but few works address the design-time question of *when agents are truly needed*. Related work such as AgentBoard Chang et al. [2024] benchmarks multi-turn agent evaluation via task decomposition and error taxonomy, aligning with STRIDE's scoring. COPPER Bo et al. [2024] introduces self-reflection via counterfactual rewards in multi-agent settings, reinforcing the role of reflection analysis in STRIDE. While frameworks address components of intelligent execution Ye and Jaques, Kapoor et al. [2024], few offer a systematic methodology for selecting the appropriate AI modality at design time.

**Benchmarks for agent performance.** A growing body of benchmarks evaluates how well agents perform specific tasks. AgentBench Xu et al. [2025], ITBench Jha et al. [2025], and ToolBench Qin et al. [2025] stress-test multi-tool reasoning and environment interaction. SWE-Bench Jimenez et al. [2023] focuses on software engineering workflows, while Gorilla Patil et al. [2024] evaluates large-scale tool invocation. HuggingGPT Shen et al. [2023] and ReAct Yao et al. [2023] integrate tool usage and reasoning traces to improve robustness. These works emphasize *performance measurement after deployment*. By contrast, STRIDE addresses the orthogonal but complementary question of *necessity at design time*: before deploying agents, can we predict whether a task truly requires them?

**Task complexity and modality selection.** Prior studies classify tasks for LLMs, assistants, or agents: agents excel at workflow decomposition but risk loops IBM [2025]; small LMs suit repetitive subtasks Belcak et al. [2025], Greyling, Cobus [2025]; and governance risks remain a concern McKinsey & Company [2025]. STRIDE formalizes these intuitions into a scoring framework that balances reasoning depth, tool needs, and state requirements.

**Task decomposition, Self-reflection and adaptive reasoning.** Decomposition is central: graph-based metrics support evaluation Gabriel et al. [2024]; TDAG automates subtasks Crispino et al. [2025]; and tool-calling studies quantify volatility from nested or parallel use Masterman [2024], factors we incorporate in the True Dynamism Score. Reflection has been explored in ARTIST Plaat et al. [2025] and MTPO Wu et al. [2025]. We instead treat reflection as a necessity criterion rather than a performance add-on.

**Industry and patents.** Frameworks such as LlamaIndex, Google ADK, and CrewAI LlamaIndex [2025] enable modular workflows, while patents from Anthropic and OpenAI Zhang et al. [2024], AFP [2025] describe autonomous travel and compliance. STRIDE differs by focusing on *design-time necessity assessment*, embedding explainability and risk-awareness into early choices.

While prior work evaluates agent capabilities post-deployment, no framework automates modality selection *at design time*. STRIDE fills this gap with task complexity scoring, variability attribution, drift monitoring, and persona-specific recommendations, uniquely addressing the question of *whether agents are needed at all* and transforming solution selection into a structured, evidence-based discipline.

## 3 Methodology

In this section, we present our end-to-end framework, STRIDE (Systematic Task Reasoning Intelligence Deployment Evaluator), for assessing whether a task requires the deployment of *agentic AI*, an *AI assistant*, or a *stateless LLM call*. STRIDE systematically evaluates ***task complexity***, ***reasoning depth***, ***tool dependencies***, ***dynamism of task***, and ***self-reflection*** requirements to provide a quantitative recommendation. Figure 1 illustrates the workflow

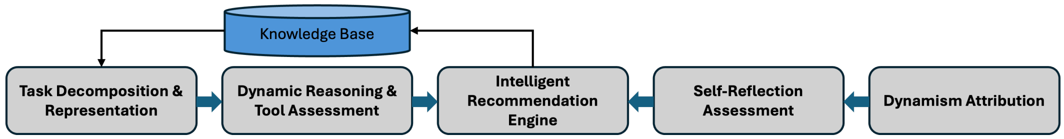


Figure 1: Overview of STRIDE, a five-stage framework for determining the necessity of Agentic AI, AI assistants, or LLM calls. Stage 1: Task decomposition into subtasks with dependency graph construction. Stage 2: Dynamic reasoning and tool-interaction scoring. Stage 3: Dynamism attribution (model/tool/workflow). Stage 4: Self-reflection requirement analysis. Stage 5: Aggregated suitability inference with persona-aware recommendations.

### 3.1 System Overview

STRIDE analyzes task descriptions, inputs/outputs, and tool dependencies to recommend the appropriate AI modality. This process comprises producing an *Agentic Suitability Score (ASS)* for each subtask. This score is then aggregated to guide the final modality recommendation:

- **Task Decomposition:** Breaks tasks into a DAG of subtasks to expose dependencies.
- **Reasoning & Tool Scoring:** Quantifies reasoning depth, tool reliance, and API orchestration requirements.
- **Dynamism Analysis:** Attributes variability across model, tool, and workflow sources using a True Dynamism Score (TDS) to determine whether adaptive agentic reasoning is needed.
- **Self-Reflection Assessment:** Detects when iterative correction is required and integrates all factors into an *Agentic Suitability Score (ASS)* to give final recommendation.

### 3.2 Task Decomposition & Representation

In this stage, STRIDE transforms free-form task descriptions into structured, actionable subtasks using a fine-tuned LLM with specialized prompting. The system identifies key action verbs (like "search," "validate," "analyze") and target nouns (such as "flights," "budget," "data") to create meaningful work units. To illustrate with a practical example, if the initial task is "Plan a 5-day travel itinerary", the Task Decomposition phase would generate subtasks like "Search Flights", "Find Hotels", "Budget Planning", and "Activity Research".

The system automatically discovers relationships between subtasks through 1) *Temporal Analysis:* Recognizing sequence requirements ("search flights before booking hotels"), 2) *Data Flow Tracking:* Identifying when one subtask's output feeds into another ("Search Flights" results inform "Budget Alerts"), and 3) *Semantic Role Labeling:* Mapping precise input/output relationships.

STRIDE creates a directed acyclic graph (DAG) where each subtask node contains, 1) *Historical Patterns:* "Search Flights" appears as the starting point in 85% of travel planning tasks, 2) *Tool Recommendations:* Proven integrations for similar subtasks, and 3) *Performance Insights:* Success rates and optimization guidance from past executions. By converting ambiguous requests into precise, interconnected subtasks, STRIDE establishes the foundation for intelligent automation decisions. This structured approach ensures no critical dependencies are missed while enabling parallel execution where possible. Let $T = \{s_1, s_2, \ldots, s_n\}$ represent the extracted subtasks, organized in graph $G = (T, E)$ where edges $E$ capture both ordering constraints and data dependencies between tasks.

### 3.3 Dynamic Reasoning & Tool Assessment

For each subtask $s_i$, STRIDE computes an *Agentic Suitability Score* (ASS) that objectively measures whether the subtask benefits from autonomous agent capabilities:

$$\text{ASS}(s_i) = w_r \cdot R(s) + w_t \cdot T(s) + w_s \cdot S(s) + w_\rho \cdot \rho(s), \tag{1}$$

where:

- $R(s)$ = Reasoning depth (0 = *Shallow*; simple lookup or direct response, 1 = *Medium*; requires comparison or basic inference, 2 = *Deep*; multi-step analysis or complex decision-making),

- $T(s)$ = tool need (0 = *None*; no external tools required, 1 = *Single*; single tool integration, 2 = *Multiple*; multiple tool orchestration needed),
- $S(s)$ = state/memory requirement (0 = *None*; stateless operation, 1 = *Ephemeral*; single session, 2 = *Persistent*;),
- $\rho(s)$ = Risk Score (compliance violations, computational Overhead, infinite loop potential).

The weighting system $(w_r, w_t, w_s, w_\rho)$ adapts to different task domains: *Reasoning-Heavy Tasks:* $w_r$) prioritizes complex multi-step tasks (e.g., $w_r = 0.4$ for itinerary planning) *Tool-Intensive Workflows:* $(w_t)$ emphasizing tasks requiring multiple tools (e.g., $w_t = 0.3$ for API-heavy workflows) *Context-Dependent Operations:* $(w_s)$ accounting for persistent context needs (e.g., $w_s = 0.2$ for multi-turn interactions) *Risk-Sensitive Applications:* $(w_\rho)$, penalizing high-risk operations (e.g., $w_\rho = 0.1$ for compliance tasks)

STRIDE continuously refines these weights through *grid search optimization* on labeled historical task data, then refines via *reinforcement learning* from deployment outcomes and *expert feedback integration* for domain-specific calibration. This scoring mechanism prevents over-engineering simple tasks with complex agentic AI solutions, while ensuring that sophisticated problems receive appropriate autonomous capabilities. The result is precise resource allocation and optimal performance across diverse task types.

### 3.4 Dynamism Attribution

Variability alone does not justify implementing AI agents. For instance, a task like *"Generate a random greeting message"* may produce different outputs each time due to model stochasticity (model-induced variability), but it can be handled effectively by a stateless LLM with temperature adjustments—no agentic autonomy is required. STRIDE distinguishes:

- *Model-induced variability*, stems from AI model limitations, including prompt ambiguity (unclear prompts causing inconsistent outputs) and stochastic randomness (probabilistic models producing different results from identical inputs). This variability typically resolves through improved prompt engineering, temperature controls, or deterministic sampling rather than requiring agentic capabilities.
- *Tool-induced variability*, arises from external dependencies, including API volatility (changing response formats, rate limits, downtime) and dynamic tool responses (varying data based on real-time conditions). These challenges typically require robust error handling, retry mechanisms, and adaptive response parsing rather than autonomous agent decision-making.
- *Workflow-induced variability*, involves systemic execution complexity, including conditional branching (different inputs triggering varied decision trees) and environmental changes (system load, user context, data availability altering optimal paths). This category most strongly indicates agentic solution needs, as it requires dynamic decision-making and adaptive planning that benefit from autonomous reasoning capabilities.

By distinguishing sources of variability, STRIDE avoids over-engineering and activates agentic AI only when autonomous reasoning materially improves task outcomes.

The *True Dynamism Score (TDS)* isolates workflow-driven variability:

$$\text{TDS}(s_i) = \alpha \cdot W(s) + \beta \cdot V(s) - \gamma \cdot M(s), \tag{2}$$

where $W(s)$ is workflow variability, $V(s)$ tool volatility, and $M(s)$ model instability. A high TDS implies that autonomy and adaptivity are required.

### 3.5 Self-Reflection Assessment

Self-reflection is required when subtasks involve mid-execution decision points or validation of nondeterministic tools.

**Mid-execution decision points** occur when workflows cannot be fully predetermined and require dynamic evaluation during execution. AI Agents implement procedural mechanisms to incorporate tool responses mid-process, while Agentic AI introduces recursive task reallocation and cross-agent messaging for emergent decision-making Sapkota et al. [2025]. These situations arise when initial

**Algorithm 1** STRIDE Scoring & Modality Inference

**Require to Input:** Task description $\tau$, knowledge base $\mathcal{K}$, thresholds $\theta, \ldots$

**Ensure to Output:** Modal suggestion $\hat{y} \in \{\texttt{LLM\_CALL}, \texttt{AI\_ASSISTANT}, \texttt{AGENTIC\_AI}\}$

Decompose $\tau$ into subtasks $T = \{s_1, \ldots, s_n\}$ and build DAG $G$
**for** each subtask $s \in T$ **do**
    Compute $R(s), T(s), S(s), \rho(s)$ and derive ASS$(s)$
    Compute $W(s), V(s), M(s)$ and derive TDS$(s)$
    Evaluate $C(s), N(s), V(s)$ to derive SR$(s)$
**end for**
Aggregate features into task profile $\mathbf{x}_T$
Return $\hat{y} = \arg\max_m f(\mathbf{x}_T; \mathcal{K})$

conditions change unexpectedly, multi-step processes reveal information influencing subsequent actions, or quality checkpoints require evaluating whether intermediate outputs meet success criteria. The Reflexion framework demonstrates how agents reflect on task feedback and maintain reflective text in episodic memory to improve subsequent decision-making Shinn et al. [2023], with studies showing significant problem-solving performance improvements (p < 0.001) Renze and Guven [2024].

**Validation of nondeterministic tools** becomes critical when working with external systems producing variable outputs. LLM-powered systems present challenges where outputs are unpredictable, requiring custom validation frameworks. This includes API responses with different data structures, LLM-generated content requiring accuracy evaluation, and web scraping tools exhibiting behavior changes due to evolving website structures. Neural network instability can lead to disparate results, requiring rigorous validation through adversarial robustness testing.

Without self-reflection, agents risk propagating errors, making incorrect assumptions about tool outputs, or failing to adapt when strategies prove insufficient. Self-reflection enables task coherence and reliability in dynamic environments. STRIDE encodes this as a decision rule:

$$\text{SR}(s) = \mathbf{1}(\text{TDS}(s) \geq \theta \wedge (C(s) \vee N(s) \vee V(s)))\,,$$

where $C(s)$ = conditional branches, $N(s)$ = nondeterministic tools, $V(s)$ = mid-execution validation, and $\theta$ = dynamism threshold. If $\text{SR}(s) = 1$, reflection hooks (e.g., error recovery, re-planning, ReAct) are triggered.

### 3.6 Intelligent Recommendation Engine

Finally, STRIDE aggregates features from subtasks into a task profile $\mathbf{x}_T$ and queries a knowledge base $\mathcal{K}$ of historical patterns. A classifier $f$ produces the final modality:

$$\hat{y} = \arg \max_{m \in \{\text{LLM}, \text{Assistant}, \text{Agent}\}} f(\mathbf{x}_T; \mathcal{K}), \quad (3)$$

with justification tailored to the user's persona (e.g., developers receive tool configurations, managers receive architectural summaries).

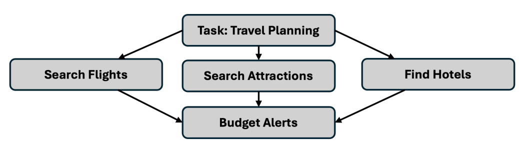


Figure 2: Toy decomposition DAG for "Plan 5-day travel itinerary." Each subtask is scored separately and orchestrated by STRIDE.

Figure 2 illustrates a toy DAG for a travel-planning task, showing how STRIDE decomposes tasks into subtasks for scoring and routing. To clarify the STRIDE workflow, Algorithm 1 outlines the end-to-end scoring and modality inference process, from task decomposition to final recommendation. This structured scoring-to-classification pipeline ensures that agentic AI is deployed only when justified by objective complexity, resource trade-offs, and dynamism.

## 4 Experiments & Results

We evaluated STRIDE across 30 real-world tasks spanning SRE, enterprise automation, legal compliance, and customer support. The objective was to test whether STRIDE reliably distinguishes

Table 2: Quantitative results of STRIDE compared to baselines across 30 tasks.

| Method | Accuracy (%) | Over-engg Reduction (%) | Resource Savings (%) |
|---|---|---|---|
| Naive Agent | 33.3 | 0 | 0 |
| Heuristic Threshold | 68.0 | 27.5 | 18.2 |
| STRIDE (ours) | **92.0** | **45.3** | **37.1** |

between LLM calls, assistants, and agents, minimizing over-engineering while ensuring accurate, cost-efficient design-time decisions. While modest in size, our task set emphasizes *depth over breadth*, demonstrating STRIDE's value in real-world settings. Across all 30 tasks, STRIDE achieved **92% accuracy**, reduced unnecessary agent deployments by **45%**, and delivered **37% lower compute/API usage** compared to always deploying agents. ***These results demonstrate that principled design-time selection yields tangible efficiency gains compared to intuition-driven deployment.*** We compared STRIDE against two baselines. The **Naive Agent** baseline always deployed agentic AI regardless of task complexity, providing an upper bound on cost but no efficiency. The **Heuristic Threshold** baseline deployed agents only when reasoning depth $\geq 2$ and tool requirements $\geq 2$, but often failed on borderline cases where task dynamism or reflection was the deciding factor. STRIDE consistently outperformed both approaches.

### 4.1 Illustrative Use Cases

To ground these aggregate results, we highlight representative tasks where STRIDE discriminates between simple lookups, medium-complexity assistance, and fully autonomous agent workflows. These cases illustrate how STRIDE's scoring pipeline translates into practical deployment recommendations.

**LLM Call Example: Currency Lookup.** *"What is the exchange rate between USD and EUR today?"* This task requires shallow reasoning (0-hop), a single API call, and no state persistence. STRIDE assigned a low True Dynamism Score (0.10) and recommended `LLM_CALL`. This minimized cost and latency, avoiding unnecessary orchestration overhead while retaining accuracy.

**AI Assistant Example: Meeting Summarization.** *"Summarize today's team meeting notes and suggest action items."* This task requires medium reasoning depth (1-hop), a summarization tool, and ephemeral state. STRIDE produced a TDS of 0.35 and recommended `AI_ASSISTANT`, reflecting that autonomy is unnecessary but structured guidance improves usability. Deploying a fully autonomous agentic AI for this task would have added unnecessary computation and orchestration overhead without improving the outcome, since an AI assistant sufficed.

**Agentic AI Example: Travel Planning.** *"Plan a 5-day travel itinerary with hotels, attractions, and budget alerts."* This task demands multi-hop reasoning, persistent state, and multiple API integrations (flights, hotels, maps). STRIDE assigned a TDS of 0.78 and correctly recommended `AGENTIC_AI`. Experts validated that dynamic replanning is essential in such workflows due to evolving constraints and interdependencies.

**SRE Example: Kubernetes Incident Analysis.** *"Analyze Kubernetes change events and correlate them with active alerts to identify the root cause of an ongoing incident."* This high-stakes task requires deep reasoning, multiple tool integrations (Kubernetes API, alerting system, causal analysis), and persistent state tracking. STRIDE scored a TDS of 0.85 and recommended `AGENTIC_AI`. Domain experts confirmed that incident resolution often requires iterative exploration and adaptive strategies that static assistants cannot provide.

**Compliance Verification Example.** *"Evaluate a set of documents for legal compliance, flagging any non-compliant sections and suggesting corrections."* This task involves deep reasoning, persistent state, and multiple specialized tools (legal database, document parser, compliance checker). STRIDE assigned a TDS of 0.80 and recommended `AGENTIC_AI`, reflecting the high compliance risks and iterative refinements required. Experts noted that assistants often fail to capture edge cases in regulatory contexts.

Table 3: Representative task evaluations. RD = Reasoning Depth, TN = Tool Needs, SN = State Needs, TDS = True Dynamism Score.

| Task | RD | TN | SN | TDS | Risk | Recommendation |
|---|---|---|---|---|---|---|
| Currency lookup | 0 | 1 | 0 | 0.10 | Low | LLM_CALL |
| Meeting summarization | 1 | 1 | 1 | 0.35 | Medium | AI_ASSISTANT |
| Travel itinerary planning | 2 | 2 | 2 | 0.78 | High | AGENTIC_AI |
| Kubernetes incident analysis | 2 | 2 | 2 | 0.85 | High | AGENTIC_AI |
| Legal compliance verification | 2 | 2 | 2 | 0.80 | High | AGENTIC_AI |

Table 4: Ablation study of STRIDE components. Accuracy, over-engineering reduction, and resource savings are reported.

| Configuration | Accuracy (%) | Over-engg Reduction (%) | Resource Savings (%) |
|---|---|---|---|
| Full STRIDE | **92.0** | **45.3** | **37.1** |
| w/o Task Decomposition | 83.0 | 35.2 | 28.0 |
| w/o True Dynamism Score | 80.0 | 33.0 | 26.5 |
| w/o TDS Weighting | 81.3 | 32.0 | 25.4 |
| w/o Self-Reflection | 76.0 | 29.5 | 22.8 |
| w/o Human-in-the-loop | 85.7 | 37.1 | 28.6 |

### 4.2 Why STRIDE Works: Ablation Study

To understand why STRIDE performs well, we conducted ablation experiments by removing core components. As Table 4 shows, each element contributes significantly. Removing task decomposition reduced accuracy by 9%, showing that subtask structure is essential for modeling dependencies. Without the True Dynamism Score, accuracy fell by 12%, as STRIDE struggled to distinguish borderline tasks like meeting summarization versus compliance verification. The largest drop came from removing self-reflection, which reduced accuracy to 76%, underscoring its role in handling mid-execution corrections and adaptive reasoning.

Human-in-the-loop validation also played a role: omitting expert feedback reduced alignment with domain judgments, demonstrating the value of incorporating expert calibration into design-time recommendations.

### 4.3 Robustness and Human Validation

Beyond aggregate numbers, we tested robustness across domains. STRIDE achieved 95% accuracy in SRE, 91% in compliance, 89% in automation, and 93% in customer support (Figure 3). This consistency suggests that STRIDE generalizes well across heterogeneous real-world tasks without overfitting to any specific domain. Errors primarily arose in borderline scenarios, such as multi-document summarization, where dynamism was underestimated. Notably, STRIDE sometimes recommended assistants when experts preferred agents, but never the reverse—avoiding costly over-engineering mistakes.

Expert validation further confirmed STRIDE's recommendations. In 78% of cases, experts fully agreed, 15% showed partial agreement (e.g., suggesting an assistant instead of an agent for borderline tasks), and only 7% disagreed (Figure 4). This resulted in a **27%** improvement in expert alignment compared to the Heuristic Threshold baseline. Feedback from engineers and compliance officers improved STRIDE through better task decomposition, adjusted TDS weights, and persona-aware outputs tailored to developers and managers (Table 5). Our robustness validation was not a one-off annotation exercise, but the result of extended collaboration with subject matter experts. For the SRE domain, three Kubernetes incident response experts engaged with STRIDE iteratively over a six-month period (March–August 2025), providing feedback on decomposition, reflection, and dynamism scoring. In the compliance domain, two legal verification experts participated in a shorter but focused engagement of 1–2 months (May-June 2025), helping calibrate task scoring against regulatory criteria. This sustained, multi-month collaboration ensured that STRIDE's assessments aligned with the nuanced realities of enterprise practice.

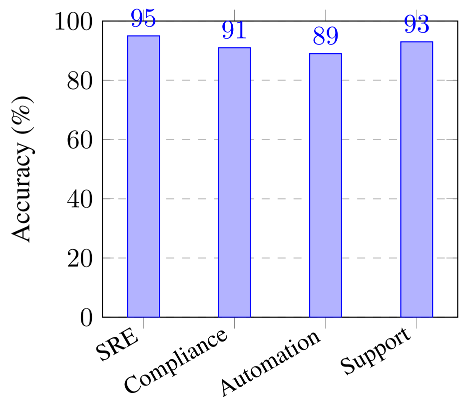


Figure 3: Domain-wise accuracy of STRIDE across 30 tasks.

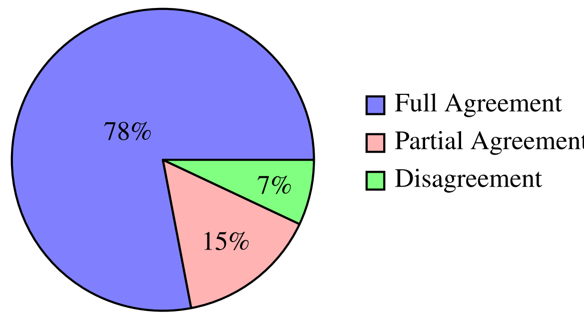


Figure 4: Expert agreement with STRIDE recommendations.

Table 5: Summary of Human-in-the-Loop Feedback and System Improvements.

| Feedback Area | Improvements Made |
|---|---|
| Task Decomposition | Enhanced LLM-driven decomposition to better capture subtask dependencies. |
| Dynamism Analysis | Adjusted weights in the True Dynamism Score to better separate model-, tool-, and workflow-induced variability. |
| Knowledge Base | Expanded task patterns and historical performance metrics for SRE and compliance tasks. |

### 4.4 Discussion and Limitations

STRIDE reduces the costs, risks, and misaligned expectations of unnecessary agents. By shifting selection to design time, it prevents over-engineering, ensures autonomy only where required, and reframes adoption from intuition-driven to structured decision process that directly translates into lower compute/API expenditure and reduced operational costs. At the same time, we acknowledge limitations. STRIDE's scoring functions are heuristic by design, striking a balance between interpretability and generality.

Finally, STRIDE complements existing benchmarks, such as AgentBench, SWE-Bench, and ToolBench. While those benchmarks evaluate *how well* agents perform after deployment, STRIDE focuses on *whether agents are needed at all* before deployment. This creates opportunities for integration: STRIDE could serve as a design-time filter that guides which tasks should be benchmarked with agents, or as a planning tool embedded into enterprise AI workflows. Together, these directions position STRIDE as both a practical engineering aid and a guardrail for responsible AI deployment.

## 5 Conclusion

We introduced STRIDE (Systematic Task Reasoning Intelligence Deployment Evaluator), a framework for systematically determining when tasks require agentic AI, AI assistants, or simple LLM calls. STRIDE integrates five analytical dimensions — structured task decomposition, dynamic reasoning and tool-interaction scoring, dynamism attribution analysis, self-reflection requirement assessment, and agentic suitability inference. In evaluating 30 real-world enterprise tasks, STRIDE reduced unnecessary agent deployments by 45%, improved expert alignment by 27% and cut resource costs by 37%, directly mitigating over-engineering risks and containing compute costs.

Looking ahead, we will extend evaluation beyond the 30 tasks to include multimodal tasks (vision/audio), integrate reinforcement learning for weight tuning, and validate STRIDE at enterprise scale. These extensions will further strengthen its role as a practical guardrail for responsible AI deployment.

## References


AFP. Openai lance un agent ia autonome qui exécute des tâches en ligne, 1 2025. URL https://apnews.com/article/nvidia-gtc-jensen-huang-ai-457e9260aa2a34c1bbcc07c98b7a0555.

Peter Belcak et al. Small language models are the future of agentic ai. *arXiv*, 6 2025. URL https://arxiv.org/abs/2506.02153.

Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. Reflective multi-agent collaboration based on large language models. *Advances in Neural Information Processing Systems*, 37:138595–138631, 2024.

Ma Chang, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

M. Crispino et al. Tdag: A multi-agent framework based on dynamic task decomposition and agent generation. *ScienceDirect*, 1 2025. URL https://www.sciencedirect.com/science/article/abs/pii/S0893608025000796.

Adrian Garret Gabriel et al. Advancing agentic systems: Dynamic task decomposition, tool integration and evaluation using novel metrics and dataset. *arXiv*, 10 2024. URL https://arxiv.org/abs/2410.22457.

Greyling, Cobus. Nvidia says small language models are the future of agentic ai, 6 2025. URL https://cobusgreyling.medium.com/nvidia-says-small-language-models-are-the-future-of-agentic-ai-f1f7289d9565.

IBM. Ai agents vs. ai assistants, 7 2025. URL https://www.ibm.com/think/topics/ai-agents-vs-ai-assistants.

Saurabh Jha, Rohan Arora, Yuji Watanabe, Takumi Yanagawa, Yinfang Chen, Jackson Clark, Bhavya Bhavya, Mudit Verma, Harshit Kumar, Hirokuni Kitahara, et al. Itbench: Evaluating ai agents across diverse real-world it automation tasks. *arXiv preprint arXiv:2502.05352*, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Sayash Kapoor, Benedikt Stroebl, Zachary S Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. *arXiv preprint arXiv:2407.01502*, 2024.

LlamaIndex. Agentic ai frameworks: Architectures, protocols, and design challenges, 8 2025. URL https://arxiv.org/html/2508.10146.

Tula Masterman. Ai agents: The intersection of tool calling and reasoning in generative ai, 10 2024. URL https://medium.com/data-science/ai-agents-the-intersection-of-tool-calling-and-reasoning-in-generative-ai-ff268eece443.

McKinsey & Company. Seizing the agentic ai advantage, 6 2025. URL https://www.mckinsey.com/capabilities/quantumblack/our-insights/seizing-the-agentic-ai-advantage.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

Aske Plaat et al. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv*, 4 2025. URL https://arxiv.org/html/2505.01441v1.

Shengqian Qin, Yakun Zhu, Linjie Mu, Shaoting Zhang, and Xiaofan Zhang. Meta-tool: Unleash open-world function calling capabilities of general-purpose large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30653–30677, 2025.

Matthew Renze and Erhan Guven. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*, 2024.

Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *arXiv preprint arXiv:2505.10468*, 2025.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Q. Wu et al. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv*, 1 2025. URL `https://arxiv.org/html/2501.09686v3`.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Eric Ye and Natasha Jaques. An efficient open world benchmark for multi-agent reinforcement learning. In *NeurIPS 2024 Workshop on Open-World Agents*.

Y. Zhang et al. Towards automated patent workflows: Ai-orchestrated multi-agent framework for intellectual property management and analysis. *arXiv*, 10 2024. URL `https://arxiv.org/html/2409.19006`.