Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Planning and Evaluation Methods in LLM-Based Autonomous Workflow Systems: A Comprehensive Review

Chanuka Bandara Wessin Arachchillage

chanuka.20222453@iit.ac.lk

Informatics Institute of Technology

Dilini Thamarasee

Informatics Institute of Technology

Prasan Yapa

Kyoto University of Advanced Science

update

Keywords: Large language models, Autonomous workflow systems, Planning methods, Evaluation frameworks

Posted Date: May 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9460995/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Planning and Evaluation Methods in LLM-Based Autonomous Workflow Systems: A Comprehensive Review

Abstract

Large language models (LLMs) increasingly serve as the cognitive core of autonomous systems that plan, reason, and execute multi-step workflows across domains such as software engineering, web automation, enterprise orchestration, and robotics. While planning methods have advanced rapidly from simple prompting strategies to multi-agent collaboration and hybrid neuro-symbolic approaches, the evaluation methodologies needed to assess these systems reliably, safely, and efficiently remain comparatively underdeveloped. Existing surveys examine agent architectures, capabilities, or collaborative patterns, but they do not systematically analyse the relationship between planning methods and the evaluation practices used to assess them. This paper presents a PRISMA-guided systematic survey of planning and evaluation methods in LLM-based autonomous workflow systems to address that gap. The review develops a five-category planning taxonomy and a five-axis evaluation framework spanning utility, efficiency, quality, robustness, and safety. A comparative analysis reveals a persistent planning evaluation misalignment: utility is the only consistently evaluated dimension across planning categories, whereas robustness, efficiency, quality, and especially safety receive far less systematic attention. Overall, the survey finds that the primary bottleneck in the field is no longer the emergence of new planning methods, but the lack of sufficiently mature evaluation infrastructure. By making this disconnect explicit, the review provides a structured foundation for future research and argues for more rigorous, multidimensional, and deployment-relevant evaluation standards for autonomous workflow systems.

Keywords: Large language models, Autonomous workflow systems, Planning methods, Evaluation frameworks

1 Introduction

LLMs have significantly reshaped the landscape of autonomous systems. Transformer-based architectures [1], scaled to unprecedented model sizes [2] and aligned through instruction tuning and reinforcement learning from human feedback [3], now increasingly serve as the cognitive core of systems that can reason, plan, and execute multi-step tasks. These systems are emerging across a wide range of domains, including

software engineering, web automation, enterprise workflow orchestration, and robotics. As this field has matured, planning methods have evolved beyond simple prompting strategies toward more structured forms of deliberation, including decomposition-based reasoning, feedback-driven adaptation, multi-agent coordination, and hybrid neuro-symbolic approaches. Yet despite this rapid methodological progress, the evaluation practices used to assess such systems remain comparatively fragmented and uneven.

Several recent surveys have examined LLM-based agents from important and complementary perspectives. [4] provide a broad architectural taxonomy of autonomous agents. [4] examine agent capabilities and broader implications. [5] focuses on multimodal interaction in agent systems. [6] emphasise collaborative workflow patterns and multi-agent coordination. While these surveys offer valuable overviews of architectures, capabilities, and application settings, they do not systematically examine how planning methods and evaluation practices relate to one another. In particular, the literature still lacks a focused synthesis of which planning approaches are being evaluated, which evaluation dimensions receive sustained attention, and where methodological misalignment persists between claimed capabilities and actual empirical assessment.

This review addresses that gap by presenting a systematic survey of planning and evaluation methods in LLM-based autonomous workflow systems. Rather than treating planning and evaluation as separate themes, the review examines them as interdependent components of system design. The goal is not only to organise the diverse planning strategies that have emerged in the literature, but also to assess how these strategies are evaluated, where current evaluation coverage is insufficient, and which unresolved issues most strongly limit rigorous comparison and dependable deployment.

This review is guided by three research questions:

1. What major planning approaches are used in LLM-based autonomous workflow systems?
2. How are these systems evaluated, including which evaluation dimensions, benchmark settings, and empirical practices receive the greatest attention?
3. To what extent are planning innovations aligned with the evaluation methodologies used to validate them, and what methodological gaps remain?

To address these questions, the review pursues five analytical objectives:

- develop a structured taxonomy of planning methods used in LLM-based autonomous workflow systems;
- synthesise the major evaluation dimensions and benchmark practices used to assess these systems;
- analyse the alignment between planning innovations and the evaluation methodologies used to validate them;
- identify recurring methodological and empirical gaps that constrain progress in the field; and
- translate these observations into a forward-looking research agenda for more robust and systematically evaluated workflow-oriented agent systems.

The remainder of this paper is organised as follows. Section 2 describes the review methodology, including the search strategy, study selection criteria, and the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guided screening process. Section 3 establishes the conceptual foundations required for the later analysis, including the transition from transformer-based language models to autonomous workflow systems. Sections 4 and 5 present the planning taxonomy and evaluation framework, respectively. Section 6 brings these strands together through comparative synthesis, including planning and evaluation alignment, cross-domain analysis, and the identification of research gaps. Finally, Section 7 concludes the review by summarising the main findings and discussing their broader implications for the field.

2 Methodology

This review follows a structured and reproducible methodology informed by the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) framework. In this work, the framework was adapted to suit the fast-moving and interdisciplinary nature of research on LLM-based autonomous workflow systems, where relevant contributions are distributed across peer-reviewed conferences, journals, and influential preprints. The methodology was designed not only to identify relevant studies systematically, but also to support the review's broader analytical goals: deriving a planning taxonomy, characterising evaluation practices, and identifying evidence grounded research gaps across the field.

2.1 Search Strategy and Scope

The literature search was conducted across major academic databases and repositories, including ACL Anthology, IEEE Xplore, Springer Link, OpenReview, arXiv, and Google Scholar as a supplementary source for citation tracking. The search covered publications from January 2017 to March 2025. The lower bound was selected to align with the introduction of the Transformer architecture [1], which established the foundation of modern LLMs.

Search queries combined terms related to three concept groups: LLM-based agents, planning, and evaluation. Keywords included terms such as "large language model", "LLM", "agent", "autonomous", "planning", "workflow", "reasoning", "evaluation", "benchmark", and "metric", using combinations across titles, abstracts, and keywords where supported.

To improve coverage, the database search was supplemented with backward and forward citation tracking from several influential survey papers in the field [4] [7] [5]. This broader strategy was necessary because relevant studies are distributed across multiple research communities and often use inconsistent terminology.

2.2 Inclusion and Exclusion Criteria

To maintain a clear and consistent review boundary, explicit inclusion and exclusion criteria were applied during the screening process. These criteria were designed to focus

the review on technically substantive studies at the intersection of LLM-based agency, planning, and evaluation, while excluding adjacent work that fell outside the analytical scope of this review. The criteria used in the screening process are summarised in Table 1.

The inclusion criteria ensured that selected studies involved LLMs as core reasoning or decision-making components within autonomous or workflow-oriented systems, and that they made a meaningful contribution to planning methods, evaluation approaches, or both. The exclusion criteria removed studies focused solely on non-agentic LLM applications, single-turn systems, classical planning without LLM integration, or papers lacking sufficient technical depth.

These criteria were applied to ensure conceptual coherence with the focus of this review, namely autonomous workflow systems that involve multi-step, goal-directed task execution supported by LLM-based planning and evaluated through substantive empirical or methodological analysis.

Table 1 Inclusion and exclusion criteria used during study screening

Criterion	Description
Inclusion Criteria	
I1. LLM-based agency	Study involves LLMs as core reasoning or decision-making components within an autonomous agent or workflow system.
I2. Planning or evaluation focus	Work addresses planning methods, evaluation approaches, or both in the context of multi-step, goal-directed task execution.
I3. Technical contribution	Study presents empirical results, substantial methodological contributions, or comprehensive analytical syntheses.
I4. Language and period	Published in English between January 2017 and March 2025.
I5. Publication type	Peer-reviewed paper or high-impact preprint with significant community adoption.
Exclusion Criteria	
E1. Non-agentic LLM work	Focused exclusively on language modelling or single-capability NLP tasks without autonomous components.
E2. Single-turn systems	Question-answering or conversational systems lacking multi-step execution or environmental interaction.
E3. Classical planning only	Symbolic reasoning or classical planning without LLM integration.
E4. Insufficient substance	Purely conceptual or opinion-based papers without technical depth or experimental validation.
E5. Duplicates	Where multiple versions exist, only the most complete or formally published version was retained.

2.3 Screening and Selection Process

Study selection followed a two-stage screening process aligned with PRISMA recommendations, consisting of screening followed by full-text assessment. The overall selection process is summarised in Figure 1.

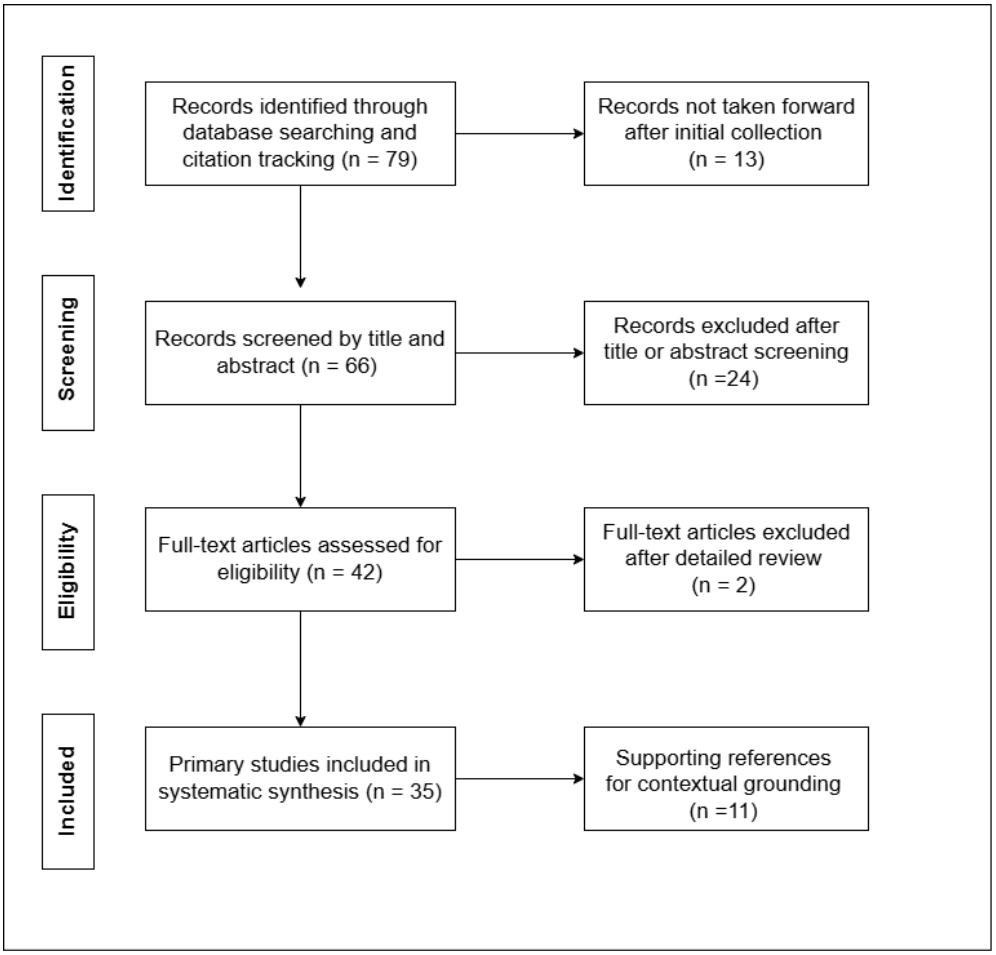


Fig. 1 PRISMA-adapted study selection flow used in this review. The diagram summarises the identification, screening, eligibility assessment, and final inclusion of primary studies and supporting references.

Following the screening and full-text review process, the final corpus comprised 79 papers. Of these, 66 were retained as primary studies for the systematic synthesis, while 13 were included as supporting references to provide foundational background, related survey context, or conceptual framing.

This distinction between primary studies and supporting references is methodologically important. The primary studies form the main evidence base for the planning taxonomy, evaluation analysis, and cross study synthesis presented in later sections, whereas the supporting references are used to strengthen conceptual grounding without being treated as direct analytical evidence in the same sense.

2.4 Data Extraction and Quality Assessment

A structured data extraction process was used to collect consistent information from each included study. For each paper, key details were recorded regarding bibliographic information, system characteristics, planning methods, evaluation approaches, and major findings. This ensured that studies could be compared systematically across both planning design and evaluation practice.

In addition to data extraction, each study was reviewed qualitatively for methodological quality. The assessment considered factors such as experimental rigour, reproducibility, breadth of evaluation, and the extent to which the reported conclusions were supported by evidence.

Studies were not excluded only on the basis of quality assessment. Instead, the quality review was used to inform the interpretation of findings during synthesis, particularly when identifying stronger evidence, methodological limitations, and recurring reproducibility concerns across the literature.

3 Foundations

This section establishes the conceptual foundations required to support the analysis presented in the later sections of this review. It is not intended to provide a comprehensive tutorial on LLMs or autonomous agents, as several high-quality surveys already serve that purpose [5] [4]. Instead, the objective here is to introduce the key concepts, terminology, and architectural distinctions necessary to interpret the planning taxonomy (Section 4), evaluation framework (Section 5), and comparative synthesis (Section 6), which together form the core analytical contributions of this paper.

For clarity, Figure 2 illustrates the conceptual role of this section within the broader review, showing how the foundations established here support the planning taxonomy (Section 4), evaluation framework (Section 5), and comparative synthesis (Section 6) developed in the subsequent sections.

3.1 Large Language Models as Reasoning Engines

The foundation of LLM-based autonomous agents lies in the transformer architecture [1], whose self-attention mechanism enables the parallel processing of sequential data and supports the modelling of long-range dependencies more effectively than earlier recurrent approaches. The rapid scaling of transformer-based language models from GPT-2's 1.5 billion parameters to GPT-3's 175 billion parameters and beyond led to qualitative changes in model behaviour that were not fully predictable from earlier scaling trends [2]. Among the most significant of these changes was the emergence of in-context learning, namely the ability to perform previously unseen tasks from a

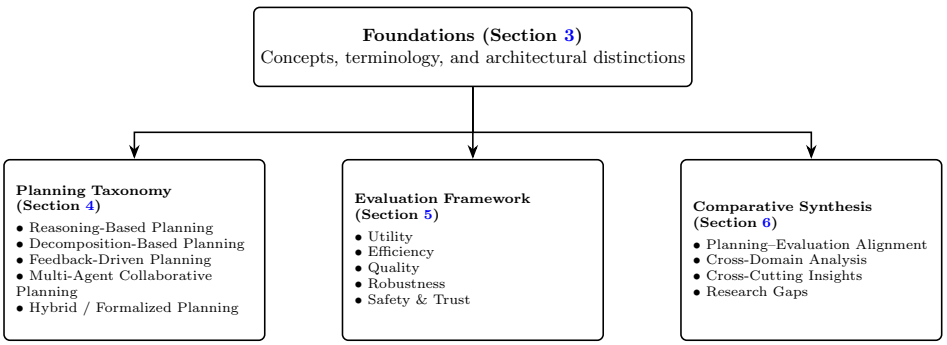


Fig. 2 Conceptual role of the foundations section in the review framework. The figure illustrates how the conceptual foundations introduced in Section 3 support the three core analytical components of the review: the planning taxonomy (Section 4), the evaluation framework (Section 5), and the comparative synthesis (Section 6).

small number of demonstrations without parameter updates [2]. This capability shifted language models from being primarily task-specific predictors toward functioning as more general-purpose reasoning systems.

Two subsequent developments were especially important in transforming LLMs from capable text generators into systems that can support structured reasoning. First, instruction tuning and reinforcement learning from human feedback (RLHF) improved alignment between model outputs and user intent, enabling models to follow complex and multi-step instructions more reliably [3]. Second, chain-of-thought [8] prompting showed that prompting models to produce intermediate reasoning steps can substantially improve performance on arithmetic, commonsense, and symbolic reasoning tasks, with this behaviour appearing as an emergent capability at sufficiently large model scales. Together, instruction-following and explicit intermediate reasoning created the practical conditions under which LLMs could begin to function as the cognitive core of autonomous agents: they could interpret goals expressed in natural language, decompose them into reasoning steps, and generate structured action sequences.

At the same time, it is important to emphasise that LLMs remain fundamentally probabilistic text generators rather than formally verified reasoning systems. Their outputs are approximate rather than guaranteed, and they remain vulnerable to hallucination, inconsistency, and context-window limitations. These constraints are not merely implementation details; they directly shape the planning methods examined in Section 4. In particular, they help explain why purely neural planning approaches often perform well in flexible reasoning settings, but why hybrid methods that incorporate formal verification or symbolic structure become necessary in domains where correctness and reliability are critical.

3.2 From Language Models to Autonomous Agents

The transition from a language model that generates text to an agent that acts in an environment requires more than simply increasing model capability. A standalone language model can reason about a task in text, but it cannot directly perceive

an environment, maintain persistent memory across interactions, or execute actions beyond token generation. By contrast, an autonomous agent operates through a broader perceive reason act cycle: it receives observations from an environment, reasons about possible actions, executes those actions through tools or effectors, and then incorporates the resulting feedback into subsequent decisions.

A particularly useful formalisation of this distinction is provided by the CoALA framework [9]. CoALA characterises an LLM-based agent as consisting of four interacting components: a core language model that serves as the reasoning engine, memory systems that support persistent storage and retrieval, an action space that defines the operations available to the agent, and a decision-making procedure that determines how observations are translated into actions. This decomposition is analytically valuable because it clarifies that many planning methods differ less in their underlying components than in how these components are orchestrated. For example, chain-of-thought [8] prompting primarily alters internal reasoning behaviour, ReAct [10] changes the sequencing of reasoning and external interaction, and multi-agent systems distribute the decision-making process across multiple cooperating entities. In each case, the same broad architectural elements remain present, but the planning mechanism changes how they are coordinated.

This distinction between LLM-as-tool and LLM-as-agent is also highly consequential for evaluation. When an LLM is used as a tool, for example, to answer a question, generate a summary, or produce a code snippet, evaluation is often relatively straightforward because the task can be framed as comparing an output to a reference or rubric. However, once the LLM operates as an agent navigating websites, debugging software repositories, or managing enterprise workflows, evaluation becomes inherently more complex. It must account not only for final outcome correctness, but also for multi-turn interaction, environmental state changes, tool-use behaviour, intermediate decision quality, and process robustness. This shift in evaluation complexity directly motivates the framework developed later in Section 5.

3.3 Autonomous Workflow Systems

The focus of this review is a specific subset of agentic systems: autonomous workflow systems. These are agent architectures that plan and execute multi-step, goal-directed task sequences in interactive environments. To distinguish them from simpler LLM applications, this review adopts four defining characteristics.

First, autonomous workflow systems involve multi-step execution. Rather than producing a single response, the system must carry out a sequence of interdependent actions. Second, they are goal-directed, meaning that these actions are organised toward the completion of a specified objective, which introduces an explicit planning requirement. Third, they are tool-augmented, interacting with external tools, APIs, databases, or execution environments rather than relying only on text generation. Fourth, they involve environment interaction, meaning that the system receives feedback from its operating environment and adapts its behaviour over time.

These four properties distinguish the systems considered in this review from both conventional LLM applications and classical autonomous systems. Conventional LLM applications typically produce one-shot outputs without persistent interaction or

external execution. Classical autonomous systems in robotics, by contrast, operate in physical environments rather than primarily digital ones, although this distinction is becoming less rigid as embodied LLM agents continue to emerge.

The systems that satisfy all four criteria already span a wide range of application domains in the recent literature. In software engineering, agents navigate complex codebases and generate patches across large repositories [11]. In web automation, agents interact with realistic browser environments to accomplish user-defined tasks [12]. In enterprise automation, agent systems coordinate actions across multiple organisational tools and business processes [13]. In embodied AI, language-guided agents plan and act within simulated or semi-physical environments [14]. More recently, multi-step scientific research assistants have begun to support tasks such as hypothesis generation, experimental design, and result interpretation.

The diversity of these domains introduces a central tension that runs throughout this review. Planning methods must be general enough to transfer across heterogeneous tasks, yet sufficiently structured to exploit domain-specific constraints and affordances. Likewise, evaluation methods must be standardised enough to support cross-system comparison, while still remaining ecologically valid enough to capture domain-specific requirements. How current work negotiates this tension is one of the recurring themes in the sections that follow.

3.4 Core Architectural Components

Across the autonomous workflow systems reviewed in Sections 4 through 7, four architectural components appear repeatedly: the core LLM, memory, tool integration, and retrieval. Understanding these components is important because they form the substrate upon which different planning methods operate.

The core LLM functions as the main reasoning engine of the system. It interprets task descriptions, generates intermediate reasoning steps, selects actions, and synthesises outputs from observations, retrieved knowledge, and tool responses. In this sense, the LLM provides the cognitive backbone of the agent, but its effectiveness depends heavily on how the surrounding architecture supports and constrains its reasoning.

Memory systems provide persistent storage beyond the immediate context window of the LLM. Following the taxonomy proposed in CoALA [9], it is useful to distinguish between working memory (the active context window), episodic memory (records of prior interactions and experiences), semantic memory (stored factual or conceptual knowledge), and procedural memory (reusable skills or action patterns). Different agent systems instantiate these forms of memory in different ways. MemGPT [15], for example, demonstrated that virtual context management analogous to operating system memory paging can support effectively unbounded conversational context by moving information between active and long-term storage. VOYAGER's skill library illustrates procedural memory in the form of reusable executable code accumulated over time [14]. Generative Agents use a memory stream and reflection-based retrieval process to support episodic recall and contextually relevant reasoning over past events [16]. These memory mechanisms are highly relevant to planning because they determine what information is available to the system at each decision point.

Tool integration enables agents to act beyond language generation. Early function calling approaches demonstrated that LLMs could invoke external operations, but later systems significantly expanded this capability through richer tool-use frameworks such as HuggingGPT [17] and Gorilla [18]. From a planning perspective, tool integration fundamentally changes the nature of the action space. An agent that can query a database, call a web API, execute code, or manipulate files faces a qualitatively different planning problem from one restricted to textual reasoning alone.

Retrieval and grounding mechanisms connect agent reasoning to external knowledge sources and help mitigate hallucination by grounding outputs in retrieved evidence. Retrieval-augmented generation (RAG) established the foundational pattern of retrieving relevant documents before generation in knowledge-intensive tasks [19]. In autonomous workflow systems, however, retrieval is not merely a support mechanism for answer quality; it can also function as a direct input to planning. Retrieved documentation, prior task traces, domain policies, or external records may all influence what actions are selected and how workflows are structured.

Finally, these components the core LLM, memory, tools, and retrieval do not operate independently. Their practical significance emerges through the planning methods examined in Section 4, which determine how they are orchestrated in a coherent workflow. The planner effectively governs which memories are consulted, what information is retrieved, which tools are invoked, and how these elements are sequenced into goal-directed action. The evaluation methods discussed in Section 5 then assess whether this orchestration produces workflows that are not only effective, but also efficient, robust, and trustworthy.

4 Planning Methods in LLM-Based Autonomous Workflow Systems

Planning is a foundational capability in autonomous workflow systems because it enables LLMs to move beyond generic reasoning and produce structured, goal directed behaviour. In classical artificial intelligence, planning is typically defined as the process of selecting a sequence of actions that transforms a system from an initial state into a desired goal state. In LLM-based autonomous systems, however, planning becomes significantly more challenging. These systems rarely operate in fully formalised environments instead, they must plan in open ended, partially observable settings where goals may be underspecified, relevant context may be incomplete, and feedback is frequently communicated through natural language rather than formal state transitions.

This section develops a systematic taxonomy of planning methods based on the primary studies reviewed in the corpus described in Section 2. From this analysis, five major categories of planning methods are identified: reasoning-based planning, decomposition-based planning, feedback-driven planning, multi-agent collaborative planning, and hybrid planning, as illustrated in Figure 3. The section concludes with a detailed analysis that highlights recurring patterns, major trade-offs, and broader

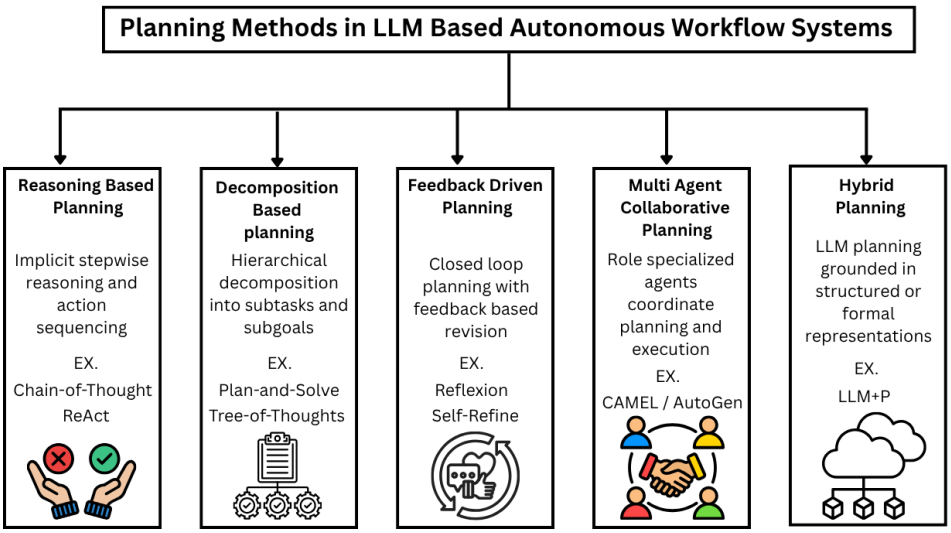


Fig. 3 Taxonomy of planning methods in LLM-based autonomous workflow systems. The taxonomy is derived inductively from the reviewed literature and groups planning approaches into five major categories based on their dominant planning mechanism. Example systems and paradigms shown in each category are illustrative rather than exhaustive.

implications across these approaches, particularly in relation to the evaluation methodologies examined in Section 5. Table 2 provides a comparative overview of these planning categories.

4.1 Reasoning-Based Planning

Reasoning-based planning methods generate plans through structured inference rather than through iterative interaction with the environment. In this category, the model produces a plan by internally organising intermediate reasoning steps, without relying on external feedback during execution. The foundational method is chain-of-thought (CoT) prompting [8], which encourages the model to produce intermediate reasoning steps before generating a final answer. This is typically achieved either by providing exemplar reasoning chains or by using simple instructions such as "let's think step by step" [20]. As a result, the model produces multi-step reasoning that has been shown to improve performance on arithmetic, commonsense, and symbolic tasks. More importantly, CoT demonstrates that natural language itself can serve as a planning medium, where each reasoning step functions as an intermediate state within a linear plan. However, CoT remains an open-loop approach and is therefore vulnerable to cascading errors, where a mistake in an early step can propagate through the remainder of the reasoning chain.

This basic mechanism was later strengthened through self-consistency decoding [21]. Instead of relying on a single reasoning chain, self-consistency generates multiple independent reasoning paths for the same problem and selects the final answer through

majority voting. This introduces a limited form of multi-path planning: rather than committing to one trajectory, the model explores several alternative reasoning paths and aggregates their outcomes. In practice, this improves robustness while preserving the simplicity of prompt-based reasoning and without requiring additional fine-tuning.

Although effective, linear reasoning chains are limited in their ability to represent branching decisions or recover from poor intermediate choices. To address this limitation, later work introduced tree and graph structured reasoning, enabling the model to explore multiple possible paths, revise earlier decisions, and combine partial insights. *Tree of Thoughts* [22] extends CoT by representing reasoning as a tree of intermediate states. At each step, the model generates several candidate thoughts, evaluates their promise using model-based heuristics, and selectively expands the most promising branches through breadth-first or depth-first search. This shifts planning from simple linear progression to deliberate exploration, allowing the model to compare alternative partial plans before committing to a final trajectory. *Graph of Thoughts* [23] generalises this further by modelling reasoning as a directed graph rather than a tree. This allows information to be reused across branches, supports iterative refinement of intermediate thoughts, and enables backtracking from unproductive paths. Compared with tree-based approaches, graph-based reasoning highlights how richer reasoning structures can improve both planning quality and computational efficiency by allowing intermediate reasoning states to be shared and revisited.

A related line of work enhances reasoning-based planning by delegating computation to external program interpreters while keeping the LLM responsible for structuring the reasoning process. *Program-Aided Language Models* [24] prompt the model to express its reasoning as executable Python code, which is then run by an external interpreter. This reduces arithmetic and symbolic errors that often arise in purely natural-language reasoning. *Program of Thoughts* prompting [25] extends this idea by incorporating richer programming constructs such as loops and conditionals. These approaches are particularly important in the broader planning literature because they establish executable code as a valid planning representation rather than merely a computation aid. This idea later becomes central in more explicitly agentic systems such as *VOYAGER* [14] and *OS-Copilot* [26], where code functions not only as the plan itself but also as a reusable skill artefact.

Overall, reasoning-based planning methods remain attractive because they are simple to implement, interpretable, and highly flexible across tasks. However, they also have important limitations. In their linear forms, they struggle to represent branching dependencies; more generally, they remain constrained by the model's context window and lack native mechanisms for adaptation when execution outcomes differ from expectation. These limitations motivate the more structured and adaptive planning approaches discussed in the following subsections.

4.2 Decomposition-Based Planning

Decomposition-based planning methods address the complexity of multi-step tasks by explicitly breaking high-level goals into smaller, structured sub-goals before execution. Unlike reasoning-based methods, which often generate a single monolithic reasoning chain, decomposition-based approaches introduce an explicit task hierarchy that

organises the planning problem into manageable components. This distinction is especially important in autonomous workflow systems, where tasks often involve multiple interdependent stages that require different tools, data sources, or reasoning strategies.

At the prompting level, one of the foundational methods in this category is least-to-most prompting [27]. This is a two-stage approach in which the model first decomposes a problem into a sequence of progressively more complex subproblems and then solves them one by one, using earlier solutions as context for later steps. The key contribution of this method is that explicit decomposition can improve compositional generalisation, allowing the model to solve problems that are more difficult than those directly represented in the prompt examples. *Decomposed Prompting* [28] extends this idea by assigning different subtasks to specialised handlers, such as distinct LLM prompts, external tools, or domain-specific modules. In this setting, the decomposer LLM acts as a planner that determines both the task structure and the appropriate allocation of resources. A particularly important insight from this work is that LLMs often fail more often in structuring the task than in solving the individual subtasks themselves, highlighting that decomposition is not merely a convenience but a central planning challenge. *Plan-and-Solve Prompting* [29] further reinforces this principle by showing that even minimal planning scaffolding such as instructing the model to first "devise a plan" before solving can outperform zero-shot chain-of-thought prompting. Taken together, these methods show that introducing an explicit planning phase, even in lightweight prompt-based forms, can improve downstream reasoning quality.

These prompting-level ideas are extended more fully in hierarchical agent architectures. *OS-Copilot* [26] implements a three-layer planning hierarchy in which a Planner first decomposes the user request into high-level subtasks, a Configurator then selects the appropriate tools for each subtask, and an Executor carries out the actual tool invocations at the operating system level. This separation of strategic, tactical, and operational responsibilities illustrates how hierarchical decomposition can improve modularity and support stronger task performance in complex environments. Similarly, *Generative Agents* [16] apply decomposition in a temporal form by constructing daily plans at a coarse level and progressively refining them into hourly and minute-level actions. This temporal hierarchy helps maintain long-term behavioural coherence while still allowing responsiveness to immediate events through memory-informed replanning.

Overall, decomposition-based methods provide a more structured way to manage task complexity than purely linear reasoning approaches. By explicitly organising tasks into subtasks, they improve modularity, compositionality, and coordination across heterogeneous resources. However, their effectiveness depends heavily on whether the model can produce an appropriate decomposition in the first place. If critical subtasks are omitted, unnecessary branches are introduced, or dependencies are ordered incorrectly, the quality of the entire plan can be compromised. In open-loop settings, such failures are often difficult to detect and even harder to recover from once execution has begun. These limitations directly motivate the feedback-driven methods discussed next.

4.3 Feedback-Driven Planning

Feedback-driven planning methods represent a fundamental departure from open-loop approaches by interleaving plan generation with execution and using the resulting observations to iteratively refine the plan. These methods are grounded in the recognition that, in realistic autonomous workflow systems, the consequences of actions cannot always be predicted reliably in advance. As a result, robust planning often requires closed-loop adaptation rather than one-shot reasoning alone.

The foundational framework in this category is *ReAct* [10], which interleaves reasoning traces with actions in an alternating thought action observation loop. At each step, the model generates a thought about the current situation, selects and executes an action, observes the result, and conditions its next thought on that observation. This design addresses a major weakness of purely reasoning-based and purely action-based approaches: reasoning without acting lacks environmental grounding, whereas acting without reasoning lacks strategic direction. Therefore the ReAct pattern become one of the most influential architectural templates in subsequent agent research, particularly in systems that rely on iterative tool use and observation-conditioned decision making.

Reflexion [30] extends this paradigm from within-episode adaptation to across-episode learning. After each task attempt, the agent generates a verbal reflection analysing its failures, stores this in episodic memory, and retrieves relevant reflections on subsequent attempts. This demonstrates that natural language can function not only as a medium for planning, but also as a mechanism for learning from planning failures without requiring model weight updates. *Inner Monologue* [31] introduces an embodied variant in which the language model receives continuous multi-channel feedback, including success detection, passive scene descriptions, and active scene queries. A particularly important insight from this work is that rich and informative feedback can substantially compensate for weaknesses in an initial plan. In practice, this suggests that effective environmental grounding may sometimes be more important than producing a highly polished open-loop plan from the outset. This observation has significant implications for workflow system design and is revisited in the comparative synthesis.

Among full agent systems, *VOYAGER* [14] provides one of the clearest demonstrations of feedback driven planning at scale. It combines an automatic curriculum that generates progressively more challenging objectives, iterative code refinement through execution, error, revision cycles, and a skill library that stores verified code for compositional reuse. This illustrates how feedback-driven planning can support not only short-term correction, but also long-term capability accumulation through reusable procedural memory. *OS-Copilot* [26] similarly incorporates execution outcomes into a broader self-improvement process that supports skill accumulation and reuse.

The strengths of feedback-driven methods including robustness, adaptability, and the capacity for experiential learning come at the cost of increased computational expense and a risk of oscillation or thrashing when feedback is ambiguous or noisy. In addition, these methods are generally easier to deploy in environments where failed actions can be retried, corrected, or safely compensated for, whereas irreversible

actions increase the cost of exploratory adaptation. These trade-offs are examined further in the comparative synthesis.

4.4 Multi-Agent Collaborative Planning

Multi-agent collaborative planning methods distribute the planning problem across multiple LLM-based agents, each with distinct roles, knowledge, or capabilities. This approach is motivated by the observation that complex workflows often require diverse forms of expertise that may exceed the ability of a single agent to maintain coherent reasoning within a limited context window. By allocating different planning or execution responsibilities across specialised agents, these systems aim to improve scalability, modularity, and task coverage.

CAMEL [32] established role-based coordination as a foundational paradigm in this area, with an AI User responsible for decomposing tasks and an AI Assistant responsible for proposing solutions through structured turn-taking. This framework demonstrated that role specialisation can improve both task decomposition and solution generation, while also revealing characteristic failure modes such as role flipping, infinite loops, and topic drift. *AutoGen* [33] generalised this idea by treating conversation itself as the computational mechanism, with flexible agent interaction patterns and structured conversational coordination. In this view, planning is not confined to a single reasoning trace, but instead emerges through iterative exchanges among specialised conversational participants.

More structured multi-agent approaches subsequently emerged through standard operating procedure pipelines. *MetaGPT* [34] models software development as a sequence of well-defined stages populated by specialised agent roles such as product manager, architect, project manager, engineer, and QA engineer. Importantly, each agent produces structured artefacts such as requirement documents, system designs, and task specifications rather than relying solely on unstructured dialogue. This document-centred coordination can reduce information loss across conversational turns and introduces stronger process discipline into the planning pipeline. *ChatDev* [35] adopts a complementary strategy through chat chains, in which specialised agents engage in phase-specific dialogues within each development stage. This illustrates an alternative design principle in which coordination remains conversational, but is constrained by stage-specific interaction patterns.

A complementary perspective is provided by *Generative Agents* [16], which shows that coordinated behaviour can sometimes emerge even without an explicit centralised coordination protocol. In that system, multiple agents independently generate memory-informed hierarchical plans, yet coherent group behaviours still arise through social interaction, local communication, and shared environmental context. This suggests that multi-agent coherence may, in some settings, emerge from memory and interaction dynamics rather than from rigid hand-engineered coordination structures alone.

The effectiveness of multi-agent collaborative planning depends critically on the design of agent roles, communication protocols, and coordination structures, all of which often need to be tailored to the target application domain. Consequently, although multi-agent systems can distribute cognitive load and enable specialisation,

they also introduce coordination overhead, communication costs, and additional failure modes that complicate both deployment and evaluation. These challenges are revisited in the evaluation framework presented in Section 5.

4.5 Hybrid and Formalized Planning

Hybrid and formalised planning methods integrate LLMs with external formal systems such as classical planners, symbolic solvers, or domain-specific languages in order to combine the flexibility of natural language reasoning with forms of correctness or consistency that purely LLM-based planning cannot reliably provide. This category directly addresses a recurring limitation in empirical studies: although LLMs can often describe plausible plans in natural language, they frequently fail to produce formally valid or executable plans for problems that classical planning or symbolic reasoning systems can solve more reliably.

A prominent example is *LLM+P* [36], in which the LLM translates natural language task descriptions into PDDL (Planning Domain Definition Language) specifications, a classical planner solves the resulting formal problem, and the LLM then translates the solution back into natural language. This architecture illustrates a central insight: LLMs can be highly effective as translators between human language and formal planning representations, even when they are unreliable as standalone planners. Several works extend this idea beyond individual task instances by using LLMs to generate complete world models or domain specifications in PDDL from natural language descriptions [37]. In these systems, the LLM supports the formal planning pipeline by helping construct the symbolic representation itself.

A related but conceptually distinct line of work is represented by *RAP* [38], which reframes LLM reasoning as a planning problem by treating the language model as both a world model and a reasoning agent while using Monte Carlo Tree Search as the planning algorithm. Rather than simply handing the problem off to an external classical planner, this approach embeds explicit search structure directly into the inference process, thereby blurring the boundary between neural reasoning and algorithmic planning.

Beyond classical planner integration, other systems formalise reasoning through symbolic solvers and declarative workflow design. *LOGIC-LM* [39] translates natural language reasoning problems into formal logic programs that are then solved by external symbolic solvers. Importantly, it also incorporates a self-refinement loop in which solver feedback is used to correct translation errors. *DSPy* [40] extends formalisation to the workflow level by allowing users to specify desired language-model pipeline behaviour through typed input-output signatures, after which the system compiles these specifications into optimised prompt programs. This is especially significant because it suggests that the design and optimisation of agent workflows can itself be made more declarative, reproducible, and partially automated.

Among the categories reviewed, hybrid methods that rely on external formal solvers or classical planners can provide the strongest correctness guarantees, provided that the formal specification is accurate. However, they also impose practical constraints. They typically require formal domain models that may not exist for many real-world applications; the translation from natural language to formal specification remains

error-prone; and classical planners generally assume well-defined state spaces, which limits applicability in open-ended environments. These correctness and flexibility trade-offs are considered further in the comparative synthesis.

4.6 Planning Representations

The format in which a plan is expressed its representation has significant implications for verifiability, executability, and composability. Planning representations therefore constitute a cross-cutting dimension that spans all five planning categories, and the reviewed literature reveals several recurring representational forms.

Natural language, used by systems such as CoT, ReAct, and many multi-agent frameworks, is immediately interpretable to humans and straightforward for LLMs to generate. However, it remains inherently ambiguous and is resistant to formal verification. Code-based or executable procedure representations, employed in systems such as *VOYAGER*, PAL, and *OS-Copilot*, provide a more explicit and directly actionable form of planning. They also support composability through skill reuse, making them particularly attractive in agentic environments where plans can be stored, refined, and recombined over time. Structured task representations, often instantiated in machine-readable formats such as JSON, are exemplified by systems such as *HuggingGPT* [17], which uses explicit task identifiers, dependencies, and arguments to orchestrate multiple tools and services.

Formal representations such as PDDL and first-order logic, used in systems such as *LLM+P* [36] and *LOGIC-LM* [39], provide the strongest support for validation, verification, and solver-based optimisation. However, they require well-defined domain models and are therefore less flexible in open-ended settings. Graph-based representations, as explored in *Graph of Thoughts* [23], can encode non-linear dependencies among reasoning states and thus better support branching, aggregation, and iterative refinement.

These representational choices reflect a fundamental trade-off between flexibility and verifiability. Increasingly, the literature points not toward a single dominant representation, but toward layered or hybrid representational strategies. For example, *VOYAGER* [14] combines natural language for high-level planning with code for executable skills, while *OS-Copilot* [26] combines natural language instructions with structured tool-facing procedures. This suggests an emerging convergence toward representation pluralism, in which different representational layers are used for different stages of the planning pipeline. The implications of representation choice for evaluation methodology are examined further in Section 5.

5 Evaluation Frameworks and Metrics for LLM-Based Workflow Systems

If planning constitutes the cognitive backbone of autonomous workflow systems, as examined in Section 4, then evaluation constitutes their empirical conscience: the methods, instruments, and standards by which the research community determines whether these systems actually work. The evaluation of LLM-based autonomous agents presents challenges that are fundamentally different from those of conventional NLP

Table 2 Comparative Analysis of Planning Method Categories

Category	Representative Methods	Core Mechanism	Feedback Role	Key Strength	Key Limitation
Reasoning-Based	CoT, Self-Consistency, ToT, GoT, PAL, PoT	Structured inference without environmental interaction	Open-loop	Simple, interpretable, no architectural changes	Cascading errors, no adaptation
Decomposition-Based	Least-to-Most, Decomposed Prompting, Plan-and-Solve, OS-Copilot, Generative Agents	Hierarchical task breakdown before execution	Minimal	Manages complexity and supports modular tool use	Incorrect decomposition is difficult to detect
Feedback-Driven	ReAct, Reflexion, Inner Monologue, VOYAGER	Interleaved planning and execution with iterative refinement	Closed-loop	Robust, adaptive, and supports experiential learning	Computationally expensive; risk of thrashing
Multi-Agent	CAMEL, AutoGen, MetaGPT, ChatDev, Generative Agents	Distributed planning through specialised roles and communication	Communication-based	Distributes cognitive load and enables specialisation	Coordination overhead; role flipping
Hybrid/Formal	LLM+P, LOGIC-LM, RAP, DSPy, World Models	Integration of LLMs with classical planners or symbolic solvers	Solver/verifier	Strongest correctness guarantees	Requires formal domain models

systems. These systems do not merely generate text; they take actions in environments, coordinate across multiple turns, invoke external tools, maintain internal state over extended horizons, and produce outcomes whose quality may depend on both objective correctness and subjective human judgment.

This section presents a systematic analysis of the evaluation landscape for LLM-based workflow systems. Drawing on benchmark papers, surveys, and system-oriented studies, the discussion is organised around five core concerns: what is being measured, how it is measured, where it is measured, what remains difficult to measure, and which new practices are emerging. The discussion therefore proceeds from core evaluation

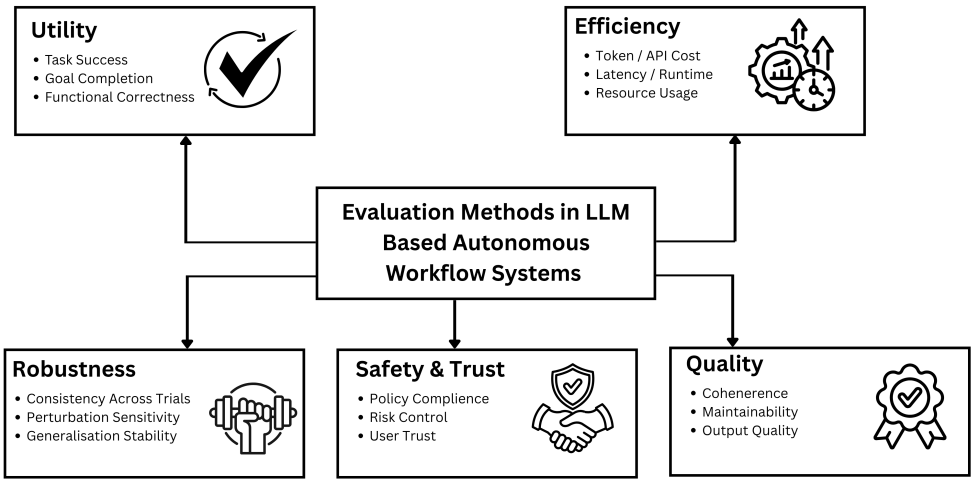


Fig. 4 Evaluation framework for LLM-based autonomous workflow systems. The figure summarises five core evaluation dimensions identified in the reviewed literature utility, efficiency, quality, robustness, and safety/trust together with representative indicators for each dimension. The listed indicators are illustrative rather than exhaustive.

dimensions to methodological approaches, benchmark ecosystems, and the principal limitations of current evaluation practice.

5.1 Core Evaluation Dimensions

Our analysis of the surveyed literature reveals five major evaluation dimensions: utility, efficiency, quality, robustness, and safety- and trust-related properties. No single study consistently addresses all five dimensions, and the uneven distribution of attention across them is itself a notable finding. Figure 4 summarises the multi-dimensional evaluation framework derived from the review and highlights representative indicators associated with each dimension.

5.1.1 Utility

Utility concerns whether a system successfully accomplishes its intended task objective. This is the most widely measured dimension in the literature and typically appears in the form of task completion rate, success rate, exact match, or environment-specific functional correctness.

Several influential benchmarks illustrate this emphasis. *AgentBench* [41] evaluates agents across multiple interactive environments and reports environment-specific task performance that is subsequently aggregated into overall benchmark comparisons, revealing a substantial performance gap between stronger commercial models and weaker open-source models. *GAIA* [42] primarily relies on strict answer correctness criteria for realistic, tool-dependent tasks and demonstrates a large gap between human performance and leading LLM systems. Similarly, *WebArena* [12] measures

whether an agent successfully completes realistic web tasks by checking the resulting state of the environment rather than comparing textual outputs, again revealing a substantial human-AI gap.

A particularly important development within utility evaluation is the growing use of execution-based verification. Instead of judging answers through surface-level text similarity, these approaches determine success by executing the generated output and verifying the actual result. *HumanEval* [43] helped establish this principle as a standard practice in modern LLM code generation evaluation by executing generated programs against unit tests. The same logic later appears in environments such as *SWE-bench* [11], which evaluates whether generated patches resolve real software issues through repository-level execution and test validation, *WebArena* [12], which checks interface and environment states, and τ-bench [44], which verifies task completion through changes in the underlying task state and domain-specific tool interactions. Compared with text-only evaluation, execution-based verification provides a much more faithful measure of whether the agent actually solved the task.

Some benchmarks also refine utility by incorporating multiple difficulty levels. For example, *GAIA* [42] shows that performance often degrades sharply as task complexity increases, indicating that raw success rate alone may conceal important differences in how systems scale across problem difficulty.

5.1.2 Efficiency

Efficiency concerns the resources required to achieve successful task completion. This includes cost, latency, number of interaction steps, tool calls, and coordination overhead. Although efficiency is highly relevant for real-world deployment, it is measured less consistently than utility.

Several studies provide useful examples. *CAMEL* [32] highlights the resource implications of extended role-based multi-agent interactions, including increased token usage and associated inference cost as dialogue length grows. *VOYAGER* [14] implicitly foregrounds efficiency concerns in long-horizon embodied settings, where iterative exploration, tool use, and skill acquisition can accumulate substantial computational and interaction overhead over time. *MINT* [45] makes an especially important contribution by analysing turn-level interaction dynamics, showing that tools and informative feedback improve task success at different rates across successive turns. This provides a much more fine-grained view of interaction efficiency than aggregate task-level scores. *AutoGen* [33] highlights an architectural dimension of efficiency: while multi-agent collaboration can simplify task decomposition and modularisation, it may also introduce additional conversational turns and coordination overhead at inference time.

Despite these advances, systematic cost-quality trade-off analysis remains relatively rare. Many papers report strong task performance without adequately discussing the cost required to achieve it, even though deployment decisions in workflow systems often depend precisely on such trade-offs.

5.1.3 Quality

Quality evaluates how well a task is performed beyond binary task success. Two systems may both complete a task, but one may do so more coherently, more maintainably, or in a more interpretable way. Depending on the application domain, quality may therefore encompass properties such as completeness, readability, maintainability, coherence, behavioural realism, and overall task refinement.

In software-oriented settings, *ChatDev* [35] evaluates generated software artefacts using multi-criteria assessments that go beyond simple task completion, including properties such as completeness, executability, consistency, and overall output quality. This offers a richer assessment than simple pass/fail success. In more socially oriented settings, *Generative Agents* [16] uses human ratings of believability and behavioural coherence, showing that architectural components such as memory and reflection can substantially improve the realism and consistency of agent behaviour.

These examples show that quality is often domain-specific and less standardised than utility. As a result, many studies rely on composite rubrics, human ratings, or task-specific proxy measures rather than universally shared metrics. Importantly, these properties are often not captured by success rates alone.

5.1.4 Robustness

Robustness refers to whether a system continues to perform reliably under varying conditions, repeated trials, or small perturbations to the task. Although robustness is crucial for real-world deployment, it remains underrepresented in much of the current evaluation literature.

One of the most significant recent contributions in this area is the pass^k metric introduced by τ-bench [44]. Unlike the standard pass@k metric, which asks whether at least one of k attempts succeeds, pass^k is designed to capture reliability across repeated trials by rewarding consistent success over multiple independent runs of the same task. This distinction is important. A model may appear capable under pass@k while still being unreliable in repeated real-world use. For workflow systems, which often require consistent operation rather than occasional success, pass^k is arguably more informative.

PlanBench [46] contributes to robustness evaluation in a different way by testing models across diverse planning domains and capabilities, showing that strong performance on one class of planning problems does not necessarily transfer reliably to others. Likewise, *MINT* [45] shows that models with strong single-turn benchmark performance do not necessarily remain reliable in multi-turn settings, where interaction depth, feedback, and error accumulation place additional stress on system behaviour. Together, these findings suggest that robustness cannot be inferred from single-shot success alone.

5.1.5 Safety, Trust, and Alignment

Safety, trust, and alignment concern whether systems behave within acceptable behavioural, policy, and risk boundaries, and whether users can rely on them appropriately in deployment. This dimension is highly important for real-world use, yet it

remains one of the least systematically evaluated. Within this dimension, safety and alignment primarily concern constraint-respecting behaviour, whereas trust concerns whether the system's behaviour is sufficiently reliable and well-calibrated for users to depend on it appropriately.

For example, τ-bench [44] distinguishes between functional success and compliant success, showing that an agent may complete a task correctly while still violating policy or business constraints. This distinction is particularly relevant in enterprise workflow settings, where correctness alone is insufficient if the system breaks procedural or regulatory rules.

Trust related concerns are also increasingly discussed in the literature, particularly in relation to whether users can appropriately rely on agent outputs under uncertainty. In practice, however, trust is often assessed only indirectly through proxy indicators such as consistency, compliance, or human ratings, rather than through standardised calibration-oriented benchmarks. More broadly, the current benchmark ecosystem remains heavily weighted toward utility, with comparatively limited and less standardised coverage of harmful actions, unsafe tool usage, policy violations, and trust-related behaviour.

The absence of standardised safety, compliance, and trust-sensitive evaluation frameworks for workflow-oriented agents therefore remains one of the most consequential shortcomings in the current literature.

5.2 Evaluation Methodologies

Beyond what is measured, it is also necessary to examine how those measurements are obtained. The surveyed literature reveals four broad methodological families: benchmark-based evaluation, simulation-based evaluation, human evaluation, and real-world deployment assessment.

5.2.1 Benchmark-Based Evaluation

Benchmark-based evaluation remains the dominant methodology. In this approach, agents are tested on standardised task sets with predefined metrics, enabling controlled comparison across models and methods. Here, benchmark-based evaluation refers to public or standardised task suites and scoring protocols, even when those benchmarks are embedded in interactive environments.

Task centric benchmarks such as *HumanEval* [43] provide precise, reproducible measurements for relatively narrow capabilities such as code synthesis. However, for workflow oriented systems, environment centric benchmarks are often more informative. *AgentBench* [41] spans multiple interactive environments, while *WebArena* [12] provides a realistic simulated web environment with long-horizon tasks. These benchmarks better reflect the stateful, interactive, and multi-step nature of real agent behaviour.

A particularly important methodological trend within this category is the growing use of execution-based verification. By checking whether the environment or system state changes as intended, benchmark-based evaluation moves beyond superficial text matching and becomes more closely aligned with practical workflow execution.

5.2.2 Simulation-Based Evaluation

Simulation-based evaluation places agents in controlled environments that approximate real-world conditions while retaining experimental control. This is particularly useful when direct deployment is impractical or risky. Unlike benchmark-based evaluation, which emphasises standardised public task suites and comparable scoring protocols, simulation-based evaluation focuses on constructing or using controlled interactive worlds to observe behaviour under richer, often longer-horizon conditions.

Generative Agents [16] is a notable example, evaluating multiple agents in a simulated social environment to study long-term behavioural coherence, coordination, and information diffusion. *VOYAGER* [14] provides another important case, operating over long durations in an embodied environment and allowing researchers to observe learning curves, skill accumulation, and cumulative capability development.

Simulation-based methods are especially valuable because they make it possible to study long-horizon interaction, emergent behaviour, adaptation, and cumulative error dynamics in ways that static or narrowly task-centric benchmarks often cannot.

5.2.3 Human Evaluation

Human evaluation remains essential for dimensions that are difficult to automate, especially coherence, naturalness, interpretability, believability, and perceived trustworthiness. Importantly, human evaluation is a methodology rather than a dimension: the same human-centred procedure may be used to assess quality, safety, trust, usability, or behavioural realism depending on the study.

For example, *Generative Agents* [16] uses Likert-style believability ratings to assess how realistic and coherent agent behaviour appears to human observers. Other studies, including role-based multi-agent frameworks such as *CAMEL* [32], complement automated metrics with human judgements or comparative preference assessments when output quality cannot be captured adequately through task success alone. In some cases, human expertise enters the evaluation pipeline more indirectly through expert-authored tasks, curated test suites, or domain-specific rubrics, even when the final scoring itself is automated.

Despite its importance, human evaluation in this area remains methodologically inconsistent. Different studies use different scales, evaluator backgrounds, instructions, and criteria, making cross-paper comparison difficult. This lack of standardisation is one reason why human-centred properties remain harder to synthesise systematically than purely functional ones.

5.2.4 Real-World Deployment Assessment

Real-world deployment assessment offers the highest ecological validity but is also the least common methodology in the current literature. These studies evaluate systems in production-like or real organisational settings, where agents interact with actual users, real tools, and naturally occurring task distributions. Explicit real-world deployment studies remain rare in the current literature, with most work stopping at benchmark or simulation-based validation rather than reporting sustained operational outcomes in live settings.

The few available studies in this category are especially valuable because they show that benchmark performance does not always translate directly to deployment performance. Distribution shift, edge cases, operational constraints, user behaviour, and long-tail failure modes often emerge only in realistic settings. As a result, deployment studies serve as an important corrective to overreliance on benchmark scores.

However, because such studies are costly, difficult to reproduce, and often domain-specific, they remain underrepresented in the literature.

5.3 Benchmark Landscape

This subsection reviews the principal benchmarks currently used to evaluate LLM-based autonomous and workflow-oriented agents. Because benchmark design varies substantially in terms of interactivity, realism, task structure, and evaluation target, no single benchmark can adequately characterise overall agent capability. Table 3 provides a comparative summary of the most relevant benchmarks discussed in this survey, including both core workflow-oriented environments and a small number of specialised supporting benchmarks that are methodologically influential.

Table 3 Representative benchmarks for LLM-based agent evaluation (ordered chronologically by year of introduction)

Benchmark	Year	Domain	Tasks	Interaction Mode	Key Metric	Illustrative Result / Notable Gap
HumanEval	2021	Code generation	164	Single-turn	pass@k	Established execution-based verification as a methodological standard
GAIA	2023	General QA + tools	466	Multi-step tool use	Exact match	Large human–AI gap, especially on harder levels
PlanBench	2023	Classical planning	600+	Single-turn	Plan accuracy	LLMs often fail on tasks classical planners solve reliably
ToolEval	2023	API tool use	16,000+	Tool invocation	Win rate	Tool-use competence can diverge from end-to-end task success
AgentBench	2024	Multi-environment	1,360+	Multi-turn interactive	Normalised score	Strong models still struggle to generalise across environments
MINT	2024	Reasoning + tool use	586	Multi-turn + feedback	Success rate per turn	Single-turn rankings do not reliably predict multi-turn performance
SWE-bench	2024	Software engineering	2,294	Code generation + test	Resolve rate	Realistic repository tasks remain far harder than synthetic coding tasks
WebArena	2024	Web navigation	812	Long-horizon web	Functional correctness	Strong models remain substantially below human performance
τ-bench	2024	Tool-agent-user	168	Multi-role interaction	pass^k	Reliability and policy compliance remain major failure points

Among the most important general interactive benchmarks, *AgentBench* [41] provides one of the broadest multi-environment evaluations, spanning settings such as operating systems, databases, knowledge graphs, games, and web interfaces. A central finding is that performance in one environment does not necessarily transfer to another, indicating that general agent competence cannot be inferred from success in any single setting. Its failure analysis also highlights long-horizon reasoning as a dominant source of error. *GAIA* [42] complements this perspective by focusing on practical intelligence through multi-step questions that require web browsing, tool use, and reasoning. Although its question-answering format is not identical to full end-to-end workflow execution, it remains valuable as a workflow-adjacent benchmark for realistic tool-grounded problem solving, especially because its difficulty levels expose how sharply performance can degrade as task complexity increases.

WebArena [12], *MINT* [45], and τ-bench [44] are especially important for workflow-oriented evaluation because they stress sustained interaction under realistic operational constraints. *WebArena* provides a simulated web environment in which agents must navigate websites and complete long-horizon tasks, with success measured through functional correctness rather than textual similarity. Reported results suggest that current models struggle with interface hallucination, error recovery, and

maintaining goal consistency over extended interaction sequences. *MINT* [45] specifically targets multi-turn interaction under different feedback conditions, including no feedback, tool-assisted feedback, and informative language feedback. One of its most important findings is that single-turn rankings do not necessarily predict multi-turn performance, which has direct implications for deployed workflow systems that depend on sustained interaction. τ-bench [44] extends this trend by evaluating tool-agent-user interaction in realistic domains and introducing pass^k as a reliability-oriented metric. Its analysis distinguishes among wrong tool arguments, rule violations, and partial completions, thereby separating functional capability from policy compliance and operational robustness.

Domain-specific benchmarks provide particularly strong stress tests for workflow planning. *SWE-bench* [11] evaluates agents on real software engineering tasks drawn from GitHub repositories and reveals a substantial gap between performance on realistic repository-level tasks and performance on synthetic coding benchmarks. Its failure analysis identifies poor context localisation, weak cross-file reasoning, brittle engineering behaviour, and limited recovery from partial failure. These weaknesses map closely to the planning limitations discussed in Section 4, especially around decomposition, feedback adaptation, and world-model construction. *PlanBench* [46] is especially relevant to this survey because it explicitly isolates planning as the capability under evaluation. Using classical planning domains, it shows that even strong LLMs can struggle on tasks that classical planners solve reliably, providing direct evidence that planning remains a genuine weakness of current LLM systems and reinforcing the relevance of hybrid and formalised approaches.

A small number of narrower benchmarks are also methodologically influential. Although not a workflow benchmark, *HumanEval* [43] remains important because it helped establish execution-based verification as a gold standard for functional evaluation, a principle that later shaped more agentic benchmarks such as *SWE-bench* [11], *WebArena* [12], and τ-bench [44]. Similarly, *ToolLLM* [47] is relevant as a specialised benchmark for API and tool-use competence, highlighting the importance of evaluating tool selection and invocation quality separately from end-to-end task completion.

Taken together, these benchmarks reveal a clear progression in the field from static, single-turn evaluation toward increasingly interactive, tool-grounded, and long-horizon assessment. As benchmark realism increases, however, performance typically declines sharply, indicating that many apparent gains on narrow tasks do not transfer cleanly to sustained workflow execution. At the same time, benchmark diversity creates a comparability problem: different environments emphasise different combinations of reasoning, tool use, planning, memory, and policy compliance, making cross-benchmark synthesis inherently difficult. This fragmentation reinforces a central conclusion of the survey: current benchmark ecosystems provide valuable but partial evidence, and stronger evaluation frameworks are still needed to connect benchmark performance to real-world reliability, safety, and trustworthiness.

5.4 Key Evaluation Challenges

Despite substantial progress, the current evaluation ecosystem still contains several structural limitations that make it difficult to assess genuine progress in workflow-capable LLM systems. Four challenges are especially significant.

5.4.1 The Capability-Reliability Disconnect

A major limitation of current evaluation practice is that many benchmarks are designed to measure whether a system can solve a task, but not whether it can do so reliably across repeated use. Metrics such as pass@k or average success rate are useful for indicating task-solving capability, but they often obscure whether success is repeatable, stable, and dependable across repeated trials.

The pass^k metric proposed in τ-bench [44] makes this distinction explicit. A model may achieve a moderate or even strong pass@1 score while still failing to reproduce correct behaviour consistently across repeated runs of the same task. For workflow systems, where users often expect repeatable and dependable operation, reliability may be more important than isolated peak performance. Yet most benchmarks still report metrics that emphasise capability over consistency.

This creates a risk that current benchmark scores may overstate deployment readiness, particularly when failure frequency, failure severity, and recovery behaviour are not explicitly evaluated.

5.4.2 Cross-Benchmark Incomparability

A second challenge is the limited comparability across benchmarks. Different benchmarks use different task distributions, success criteria, metrics, environments, and evaluation protocols. As a result, scores reported on different benchmarks often cannot be meaningfully compared without substantial qualification.

For example, a success rate in *WebArena* [12], a pass@k score in *HumanEval* [43], and a normalised score in *AgentBench* [41] each capture different aspects of system performance under very different conditions. This makes it difficult to construct a unified picture of capability across the literature, even when the same model families are being evaluated.

Table 4 provides an interpretive summary of this issue by mapping benchmark coverage onto the planning categories introduced in Section 4. Because no benchmark explicitly labels itself by planning category, the matrix should be understood as an analytical synthesis based on the dominant planning behaviours each benchmark is most capable of exposing, rather than as a strict or exhaustive taxonomy.

The matrix suggests a clear asymmetry: feedback-driven methods appear to receive comparatively broad evaluation coverage, whereas multi-agent and hybrid or formal planning approaches remain less consistently represented across widely used benchmarks. This limits the strength of comparative claims across planning categories and reinforces the need for more balanced, planning-aware evaluation design.

Table 4 Planning Method × Benchmark Coverage Matrix

Planning Category	AgentBench	GAIA	SWE-bench	WebArena	MINT	τ-bench	PlanBench
Reasoning-Based	●	●	⊙	⊙	●	⊙	●
Decomposition-Based	⊙	●	⊙	⊙	⊙	⊙	●
Feedback-Driven	●	●	●	●	●	●	⊙
Multi-Agent	⊙	⊙	⊙	⊙	⊙	⊙	○
Hybrid/Formal	⊙	○	○	○	○	⊙	●

● = strongly represented; ⊙ = partially represented; ○ = weakly or not explicitly represented

5.4.3 The Lack of Longitudinal Evaluation

A third challenge is the scarcity of longitudinal evaluation. Most benchmarks assess agents on isolated episodes with clear start and end points. However, many workflow systems are intended to operate over longer time horizons, where they must retain context, learn from repeated interactions, adapt to changing conditions, and maintain performance stability. Importantly, longitudinal evaluation is not simply a matter of longer individual tasks; it concerns whether behaviour remains effective, stable, and adaptive across repeated or ongoing use over time.

Only a small number of studies provide meaningful evidence in this direction. *VOYAGER* [14] is a prominent example, demonstrating cumulative skill acquisition over extended operation. Some interactive system papers, such as *OS-Copilot* [26], also provide partial evidence by examining performance across repeated task exposure or iterative interaction, although such evaluations are usually narrower than fully longitudinal studies. These examples suggest that longitudinal evaluation is both feasible and informative, yet it remains rare in the broader benchmark ecosystem.

Without such evaluation, the field lacks strong evidence about cumulative learning, stability over time, degradation under drift, and the persistence or amplification of failure modes during sustained use.

5.4.4 The Process Outcome Gap

A fourth challenge is the gap between outcome-based and process-based evaluation. Most benchmarks focus primarily on whether a task was completed successfully, rather than on whether the underlying reasoning, planning, and execution process was internally sound.

This is problematic because a system may reach a correct outcome through flawed, brittle, or even hallucinated intermediate reasoning and execution. Such systems may still receive full credit in outcome based evaluation, even though they are likely to be more fragile under distribution shift and less suitable for settings where interpretability, auditability, or procedural compliance matters.

This issue is especially important in workflow systems, where planning traces, tool invocation decisions, and recovery strategies may all influence whether the system is safe, explainable, and reusable. As noted in Section 4, a system may compensate

for weak initial planning through strong feedback-based correction, but outcome-only evaluation often obscures whether success arose from robust planning, effective recovery, or merely incidental error compensation.

5.5 Emerging Evaluation Practices

Recent work points toward several emerging practices that may help move the field from narrow benchmark scoring toward more mature and deployment-relevant evaluation.

First, there is a growing move toward multi-dimensional evaluation, in which utility, efficiency, robustness, and safety are treated as complementary rather than reducible to a single score. This reflects an emerging consensus that workflow-capable agents require performance profiles rather than one-dimensional rankings.

Second, compositional and ablation-based evaluation is becoming increasingly important. Studies such as *Generative Agents* [16] and *VOYAGER* [14] show that ablation can reveal the contribution of individual architectural components, such as memory, reflection, or curriculum learning. This is particularly valuable for the planning methods reviewed in Section 4, since many systems combine multiple planning strategies and their relative contributions are otherwise difficult to isolate.

Third, failure taxonomy and structured error analysis are increasingly supplementing aggregate scores. Rather than reporting only overall success, studies such as *SWE-bench* [11], *AgentBench* [41], and some interactive system studies such as *OS-Copilot* [26] classify failures into categories such as planning errors, execution errors, context localisation failures, or recovery failures. This provides much more actionable insight for system improvement.

Finally, reliability-oriented evaluation is beginning to receive more attention. Metrics such as pass^k, repeated-trial consistency, robustness under rephrasing, and calibrated uncertainty awareness all move the field closer to assessing deployment readiness rather than benchmark-only capability.

Taken together, these emerging practices suggest a broader methodological shift: evaluation is gradually moving away from isolated scalar scores and toward richer, more diagnostic frameworks that better reflect the operational demands of real workflow-oriented systems.

6 Comparative Synthesis and Cross-Domain Analysis

The preceding sections examined planning methods and evaluation methods as largely separate bodies of work. This separation reflects not only the organisation of the present survey but also a broader pattern in the literature itself: planning innovations and evaluation innovations have often developed in parallel, with limited systematic attention to how well they align. This section brings these two strands together through a comparative synthesis that the literature has only rarely attempted explicitly. Specifically, it analyses which planning methods receive which forms of evaluation, compares how planning and evaluation practices vary across application domains, identifies evidence-grounded research gaps emerging from this analysis, and distils higher-level findings that are difficult to observe when planning and evaluation are considered in

isolation. In this sense, the section constitutes the primary analytical contribution of the survey.

6.1 Planning & Evaluation Alignment Analysis

The major question motivating this analysis is whether current evaluation methods adequately test the properties that different planning approaches claim to provide. If a planning method claims to improve multi-step reasoning, is it evaluated on genuinely multi-step tasks? If a multi-agent architecture claims to improve coordination efficiency, is coordination efficiency actually measured? These questions are especially important because planning categories are often motivated by specific intended advantages, yet evaluation frequently remains concentrated on generic task success. To examine this issue systematically, Table 5 maps the planning categories introduced in Section 4 against the evaluation dimensions synthesised in Section 5.

As in Section 4, these planning categories are analytic groupings rather than mutually exclusive ontological classes; several systems exhibit characteristics spanning more than one category. Likewise, the coverage labels in Table 5 should be interpreted as a qualitative synthesis rather than a formal meta-analytic count. They reflect the relative prominence of each evaluation dimension within the representative studies associated with each planning category, based on the coding framework developed in Sections 4 and 5.

Table 5 Planning Method × Evaluation Dimension Coverage

Planning Category	Utility	Efficiency	Quality	Robustness	Safety / Trust
Reasoning-Based (CoT, ToT, GoT, PAL)	•	⊙	○	⊙	○
Decomposition-Based (LtM, DP, Plan-and-Solve)	•	○	○	○	○
Feedback-Driven (ReAct, Reflexion, VOYAGER)	•	⊙	⊙	⊙	○
Multi-Agent (CAMEL, AutoGen, MetaGPT, ChatDev)	•	○	⊙	○	○
Hybrid/Formal (LLM+P, LOGIC-LM, DSPy)	•	○	○	⊙	○

• = substantial coverage, ⊙ = partial coverage, ○ = limited or no coverage

The matrix reveals a striking pattern: utility is the only evaluation dimension that receives broad coverage across all planning categories. Every other dimension-efficiency, quality, robustness, and safety or trust-related properties is at best partially covered and is frequently absent altogether. Utility dominates largely because it is the easiest dimension to operationalise: task success rate, pass@k, exact match, and functional correctness are comparatively well defined and often automatable. Reasoning-based methods are evaluated extensively on mathematical and logical tasks, feedback-driven methods are commonly assessed in interactive environments, and multi-agent systems frequently report task completion rates or aggregate output quality. This abundance of utility-oriented evidence enables broad cross-method comparison, but it also obscures whether utility alone is sufficient for comparing planning approaches that claim advantages along fundamentally different dimensions.

Efficiency is measured only sporadically. Representative examples include reported monetary or computational cost in systems such as *VOYAGER* [14] and *CAMEL*

[32], as well as per-turn improvement analysis in *MINT* [45]. However, the surveyed literature rarely offers systematic comparisons of cost-quality trade-offs across planning categories. Quality beyond binary success is measured primarily as output or behavioural quality in multi-agent and social-agent settings, such as *Generative Agents*' [16] believability assessments or *ChatDev*'s [35] composite software quality metrics, whereas reasoning-based and decomposition-based methods are rarely evaluated on the quality of their reasoning traces or decomposition structures themselves. Robustness receives partial coverage through metrics such as pass^k in τ-bench [44] and through systematic task variation in *PlanBench* [46], yet multi-agent systems are rarely tested for robustness to role instability, communication breakdown, or coordination failure, despite these being commonly reported failure modes. Safety remains the most critically underdeveloped dimension: apart from limited policy-compliance distinctions in τ-bench [44], no planning category is systematically evaluated for whether it reduces or increases the likelihood of unsafe, non-compliant, or otherwise harmful actions.

Several method-specific gaps deserve particular emphasis. Decomposition quality is almost never evaluated as an explicit dimension: a decomposition that yields the correct final answer through a flawed or inefficient structure typically receives the same score as one that decomposes the task well. Multi-agent coordination lacks a standardised benchmark, which makes cross-system comparison more suggestive than conclusive. Hybrid and formal methods such as *LLM+P* [36] are often motivated by the prospect of stronger formal guarantees through classical planner integration, such as validity or, under appropriate planner settings, solution optimality. However, these properties are seldom evaluated as distinct end-to-end targets. In practice, evaluation typically confirms only that the system succeeds on the task, not whether the claimed formal properties continue to hold under imperfect domain specifications, translation errors, or partial modelling failures.

Taken together, this alignment analysis suggests that the field has developed much stronger evidence for whether planning methods can produce successful outcomes than for whether they deliver the specific advantages they are often claimed to provide. This mismatch between claimed planning benefits and observed evaluation coverage represents one of the most important structural weaknesses in the current literature.

6.2 Cross-Domain Comparison

The surveyed literature is distributed unevenly across application domains, and the dominant planning and evaluation practices vary substantially across them. Table 6 summarises the principal patterns observed across the major domains represented in the corpus. As with the alignment matrix in Section 6.1, the categories in this table should be interpreted as a qualitative synthesis of dominant tendencies rather than as rigid or exhaustive classifications.

Three cross-domain patterns emerge from this comparison, each carrying important implications for method transfer and system design. First, iterative feedback appears to be broadly beneficial across domains, but its concrete implementation is highly domain-specific. In software engineering, feedback is often provided through compilation, test execution, and runtime errors. In web interaction, feedback is grounded

Table 6 Cross-domain comparison of planning and evaluation practices in the surveyed literature

Domain	Dominant Planning Style	Dominant Evaluation	Key Challenge	Evaluation Maturity
Software Engineering	Decomposition + feedback + multi-agent	Execution-based (test suites, pass@k)	Context localisation in large codebases	High
Web Interaction	Feedback-driven (ReAct-style)	Functional correctness (DOM state)	Long-horizon consistency, hallucination	High
Embodied / Robotics	Feedback-driven + code-as-plan	Simulation-based (success rate, reward)	Sim-to-real transfer, safety	Medium
Enterprise / Business	SOP-based + decomposition	Case studies, task completion	Legacy integration, standardisation	Low
General-Purpose Agents	Mixed (multi-environment)	Benchmark-based (multi-environment)	Cross-environment generalisation	Medium
Human-AI Collaboration	Two-phase (plan-then-execute)	User studies (trust, team performance)	Trust calibration, intervention design	Emerging

in DOM state changes and interface accessibility. In embodied systems, it emerges through rewards, sensor states, and environmental consequences. In enterprise automation, feedback often takes the form of user correction, process validation, or operational monitoring. The underlying mechanism execute, observe, adapt is therefore recurrent across domains, yet the literature still lacks a generalised cross-domain framework for feedback-driven planning that abstracts across these domain-specific feedback modalities.

Second, multi-step task decomposition emerges as a recurring challenge across most major domains, yet each domain tends to develop its own decomposition strategies in relative isolation. Methods originating in mathematical or symbolic reasoning are rarely examined for transfer into enterprise workflow decomposition, and software-inspired structured operating procedures are rarely evaluated as scaffolds for embodied or interactive task planning. This suggests that decomposition is not merely a method family but also a potentially transferable design principle whose cross-domain portability remains largely unexplored.

Third, domain progress appears to correlate more strongly with evaluation standardisation than with planning sophistication alone. Software engineering, for example, benefits from comparatively well-operationalised task structures and highly standardised evaluation through benchmarks such as *HumanEval* [43] and *SWE-bench* [11]. By contrast, enterprise automation frequently presents complex planning requirements but lacks any broadly accepted evaluation benchmark ecosystem. This asymmetry suggests that progress in a domain is shaped not only by the sophistication of its planning architectures but also by the availability of stable, comparable, and reproducible evaluation infrastructure.

A particularly important observation is that human-AI collaborative planning introduces evaluation dimensions that are largely absent from other domains. Emerging work on human-AI collaborative planning, including approaches such as *Plan-Then-Execute* [48], suggests that transparency of the planning process can materially influence user trust and that user involvement before execution becomes especially important in higher-risk tasks. These findings are directly relevant to deployed workflow systems, yet trust and collaboration metrics remain largely underrepresented in standardised agent benchmark ecosystems. This creates a systematic misalignment between what benchmark ecosystems currently measure and what real deployment settings often require.

Taken together, the cross-domain evidence suggests that planning innovations are often more transferable than evaluation practices. While domains repeatedly converge on similar planning motifs such as decomposition, feedback, and staged execution their evaluation ecosystems remain fragmented, domain-specific, and unevenly mature. This fragmentation makes cross-domain learning more difficult and reinforces the need

for evaluation frameworks that can generalise across application settings while still respecting domain-specific constraints.

6.3 Method Trade-Off Analysis

The surveyed literature suggests three recurring trade-offs that practitioners and researchers must often navigate when selecting planning approaches for autonomous workflow systems. These trade-offs do not imply that one method family is uniformly superior to another; rather, they highlight why planning-method choice is highly contingent on deployment constraints, evaluation priorities, and the structure of the target task.

The first is a *performance-cost trade-off*. This is the most consistently visible trade-off in the literature, yet it remains among the least systematically quantified. Within reasoning-based planning, methods such as Self-Consistency [21], Tree-of-Thought [49], and Graph-of-Thought [23] often improve performance over simpler linear prompting strategies such as Chain-of-Thought [8], albeit at increased computational cost. More generally, stronger search, branching, or ensemble-style reasoning tends to require more model calls, more tokens, and longer inference time. Although some methods partially improve this frontier through more structured search or pruning, the broader pattern of cost escalation remains. Across planning categories, however, the performance-cost frontier remains largely uncharted. For example, the literature still provides little direct evidence on whether a multi-agent system with several specialised roles is more cost-effective than a single-agent system with several feedback iterations under an equivalent computational budget. This gap is especially important because relatively few papers in the broader corpus report cost metrics consistently, making informed method selection under budget constraints difficult.

The second is a *flexibility-verifiability trade-off*. Natural-language plans, such as Chain-of-Thought [8] traces, ReAct [10] trajectories, or multi-agent conversations, are highly flexible but inherently difficult to verify in a rigorous or automated manner. Code-based plans, including approaches such as PAL [24], *VOYAGER*'s [14] skill library, or executable procedure-oriented systems such as *OS-Copilot* [26], occupy a middle ground: they are executable and partially verifiable through testing, but they do not typically provide formal guarantees. Formal or symbolic plans, such as PDDL-based pipelines in *LLM+P* [36] or symbolic intermediate representations in *LOGIC-LM* [39], provide the strongest verifiability but are restricted to more structured and well-specified domains. *DSPy* [40] points to an adjacent intermediate design space in which declarative pipeline specification enables systematic optimisation and partial evaluation without requiring full symbolic formalisation. The practical implication is that planning-method selection should be aligned with the verification requirements of the deployment context: safety-critical or auditable workflows often favour hybrid or formal methods despite reduced flexibility, whereas exploratory or loosely structured workflows benefit from more flexible planning even when formal guarantees are unavailable.

The third is a *single-agent depth vs multi-agent coordination trade-off*. The evidence here is mixed. On the one hand, multi-agent systems can provide substantial gains through role specialisation, structured collaboration, and division of labour. In

software-oriented settings, systems such as *ChatDev* [35] or *MetaGPT* [34] report strong gains in output quality relative to simpler baselines. On the other hand, multi-agent systems also introduce coordination overhead, role instability, communication failure, and unstable convergence dynamics. Systems such as *CAMEL* [32] explicitly document role-flipping and related coordination pathologies, yet the benchmark ecosystem still provides almost no controlled comparisons between multi-agent and single-agent systems under equivalent computational budgets. As a result, the boundary between tasks that genuinely benefit from multi-agent specialisation and those for which multi-agent overhead is unjustified remains poorly characterised.

Taken together, these trade-offs suggest that planning-method selection in workflow-oriented systems should be understood less as a search for a universally best architecture and more as a constrained design choice. Methods that improve performance may incur prohibitive cost; methods that maximise flexibility may weaken verifiability; and methods that increase collaborative breadth may introduce coordination burdens that offset their gains. This helps explain why the literature continues to produce diverse planning architectures without clear convergence on a single dominant paradigm, and it reinforces the need for evaluation frameworks that can compare such trade-offs explicitly rather than only reporting aggregate success.

6.4 Emergent System-Level Patterns

Two findings emerge from the comparative synthesis that are rarely articulated explicitly in individual papers but become visible when planning and evaluation evidence are examined jointly.

The first is the *hallucination cascading problem*. Each planning family exhibits characteristic failure modes that may compound rather than cancel when methods are combined or when systems operate over longer horizons. Reasoning-based planning can hallucinate intermediate steps while still producing correct final answers, creating process failures that remain invisible under outcome-only evaluation. Decomposition-based planning can generate flawed task structures that omit critical subtasks or introduce false dependencies, thereby propagating errors to downstream stages. Multi-agent systems can amplify hallucinations through social reinforcement, whereby a confident but incorrect assertion from one agent is adopted and further elaborated by others. Hybrid or formal methods can also fail in a distinctive way when they generate domain specifications that are internally consistent but factually incorrect, leading downstream planners to produce plans that are formally valid yet practically wrong. The broader implication is that the reliability of a complex workflow system may be bounded by its least reliable component, while interactions among components can create emergent failure modes that component-level evaluation alone cannot detect. Current benchmark ecosystems rarely evaluate this cascading failure property explicitly.

The second is the *feedback quality revelation*. Across the surveyed literature, the quality and richness of the feedback mechanism often appears to be at least as important as the sophistication of the initial planning method, particularly in open-ended or partially observable environments. Several influential systems suggest this pattern. Embodied and interactive systems such as *Inner Monologue* [31] and *VOYAGER* [14]

show that repeated execution error revision cycles can substantially improve performance over initial attempts. *Reflexion* demonstrates that verbal self-reflection can improve the effectiveness of a wide range of underlying planning strategies. *MINT* [45] further indicates that both tool feedback and language feedback can produce compounding gains across turns. A notable exception is found in hybrid or formal pipelines such as *LLM+P* [36], where a sufficiently strong external planner can sometimes succeed without iterative feedback, although this advantage appears most applicable in structured, well-specified, and fully observable domains. For the open-ended and partially observable environments that characterise many real workflows, the broader evidence suggests that investment in feedback infrastructure including environmental observation, self-assessment, human correction channels, and structured replanning may often yield greater practical benefit than focusing exclusively on stronger initial plan generation.

6.5 Summary

The comparative synthesis in this section highlights a field that is advancing rapidly in the design of individual planning methods, yet remains constrained by a comparatively underdeveloped evaluation ecosystem. Across the surveyed literature, planning and evaluation alignment remains uneven, with utility oriented task success still receiving far more consistent attention than quality, robustness, safety, trust, or process level assessment. Cross domain comparison further suggests that differences in apparent maturity are shaped at least as much by the availability of standardized benchmarks and evaluation protocols as by the sophistication of the underlying planning methods themselves. At the methodological level, the literature continues to expose unresolved trade-offs between performance and cost, flexibility and verifiability, and single-agent depth versus multi-agent breadth, all of which complicate method selection in practical deployments. Two broader cross cutting insights are especially significant: first, that failures can propagate across planning components as cascading hallucinations in compositional systems, and second, that in open ended or partially observable environments, the richness of feedback often appears to matter more than the sophistication of the initial plan. Taken together, these findings suggest that the field is approaching a methodological inflection point: future progress is unlikely to be determined primarily by the invention of additional planning architectures, but by whether the community can build evaluation frameworks capable of distinguishing genuinely reliable workflow competence from superficially strong task performance. In particular, the next phase of research should move beyond outcome only benchmarking toward multidimensional and deployment-relevant evaluation that captures robustness under perturbation, intermediate decision quality, safety and policy compliance, and the calibration of user trust. From the perspective of this review, the most valuable future contributions may therefore come not from making agents appear more capable, but from making their capabilities more measurable, comparable, and dependable.

7 Conclusion

This paper presents a systematic survey of planning and evaluation methods in LLM-based autonomous workflow systems. The review is motivated by a central observation: planning methods and evaluation practices in this field advance unevenly, with substantial innovation in agent planning architectures but comparatively limited progress in the methodologies used to assess them. This imbalance creates an important gap in the literature, as increasingly sophisticated workflow-oriented systems are often evaluated through narrow or incomplete empirical lenses.

The analysis yields several core findings. First, the reviewed literature reveals a broad and evolving landscape of planning approaches, spanning reasoning-based, decomposition-based, feedback-driven, multi-agent collaborative, and hybrid or formalised methods. Taken together, these approaches reflect a clear transition from relatively simple, open-loop prompting strategies toward more adaptive, interactive, and structurally complex forms of planning. This shift is particularly important because autonomous workflow systems increasingly operate in open-ended and partially observable environments, where static reasoning alone is rarely sufficient.

Second, the review shows that evaluation practices remain significantly less mature than planning methods. Across the literature, utility-oriented evaluation dominates, while robustness, efficiency, quality, and especially safety receive far less systematic attention. As a result, current evaluation practice is often sufficient to show whether a system can complete a task under favourable conditions, but insufficient to determine whether it can do so reliably, safely, or in a way that supports dependable real-world use. This planning–evaluation misalignment emerges as the central structural issue in the field.

A further contribution of the review is the identification of broader cross-cutting patterns that become visible only when planning and evaluation are analysed together. In particular, the literature suggests that failures in compositional systems may accumulate rather than cancel, meaning that weaknesses in reasoning, decomposition, coordination, or tool use can interact to create larger systemic risks. At the same time, the surveyed evidence indicates that the quality of feedback mechanisms often has a stronger influence on overall system performance than the sophistication of the initial planning strategy alone. These observations have important implications for future system design, especially in environments where adaptation and recovery matter more than one-shot planning quality.

More broadly, the review highlights that the field's primary bottleneck is no longer the emergence of new planning methods, but the absence of sufficiently mature evaluation infrastructure. Although planning capabilities expand rapidly, the dominant evaluation culture remains heavily centred on task success, with limited ability to assess reliability, stability, safety, or process quality in a systematic way. Until evaluation methods become better aligned with the true demands of autonomous workflow systems, progress remains difficult to compare rigorously and even harder to translate into trustworthy deployment.

Overall, this survey argues that future progress in LLM-based autonomous workflow systems depends not only on better planners, but on better ways of evaluating planners. By making the relationship between planning design and evaluation practice

explicit, this review provides a clearer foundation for future research and encourages a shift toward more rigorous, multidimensional, and deployment-relevant evaluation standards.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[2] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., *et al.*: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)

[3] Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al.: Training a helpful and harmless assistant with reinforcement learning from human feedback. arXiv preprint arXiv:2204.05862 (2022)

[4] Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., *et al.*: A survey on large language model based autonomous agents. Frontiers of Computer Science **18**(6), 186345 (2024)

[5] Durante, Z., Huang, Q., Wake, N., Gong, R., Park, J.S., Sarkar, B., Taori, R., Noda, Y., Terzopoulos, D., Choi, Y., et al.: Agent ai: Surveying the horizons of multimodal interaction. arXiv preprint arXiv:2401.03568 (2024)

[6] Tran, K.-T., Dao, D., Nguyen, M.-D., Pham, Q.-V., O'Sullivan, B., Nguyen, H.D.: Multi-agent collaboration mechanisms: A survey of llms. arXiv preprint arXiv:2501.06322 (2025)

[7] Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., *et al.*: The rise and potential of large language model based agents: A survey. Science China Information Sciences **68**(2), 121101 (2025)

[8] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., *et al.*: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022)

[9] Sumers, T., Yao, S., Narasimhan, K.R., Griffiths, T.L.: Cognitive architectures for language agents. Transactions on Machine Learning Research (2023)

[10] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: React: Synergizing reasoning and acting in language models. In: The Eleventh International Conference on Learning Representations (2022)

[11] Jimenez, C.E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., Narasimhan, K.: Swe-bench: Can language models resolve real-world github issues? arXiv preprint arXiv:2310.06770 (2023)

[12] Zhou, S., Xu, F.F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., et al.: Webarena: A realistic web environment for building autonomous agents. arXiv preprint arXiv:2307.13854 (2023)

[13] Wornow, M., Narayan, A., Opsahl-Ong, K., McIntyre, Q., Shah, N.H., Re, C.: Automating the enterprise with foundation models. arXiv preprint arXiv:2405.03710 (2024)

[14] Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., Anandkumar, A.: Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291 (2023)

[15] Packer, C., Fang, V., Patil, S., Lin, K., Wooders, S., Gonzalez, J.: Memgpt: towards llms as operating systems. (2023)

[16] Park, J.S., O'Brien, J., Cai, C.J., Morris, M.R., Liang, P., Bernstein, M.S.: Generative agents: Interactive simulacra of human behavior. In: Proceedings of the 36th Annual Acm Symposium on User Interface Software and Technology, pp. 1–22 (2023)

[17] Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. Advances in Neural Information Processing Systems **36**, 38154–38180 (2023)

[18] Patil, S.G., Zhang, T., Wang, X., Gonzalez, J.E.: Gorilla: Large language model connected with massive apis. Advances in Neural Information Processing Systems **37**, 126544–126565 (2024)

[19] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., *et al.*: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems **33**, 9459–9474 (2020)

[20] Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. Advances in neural information processing systems **35**, 22199–22213 (2022)

[21] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)

[22] Muralidharan, J., Thomas, T.: Deliberate problem-solving with a large language model as a brainstorm aid using a checklist for prompt generation. The Journal

of the Association of Physicians of India **72**(5), 89–90 (2024)

[23] Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., *et al.*: Graph of thoughts: Solving elaborate problems with large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 17682–17690 (2024)

[24] Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., Neubig, G.: Pal: Program-aided language models. In: International Conference on Machine Learning, pp. 10764–10799 (2023). PMLR

[25] Chen, W., Ma, X., Wang, X., Cohen, W.W.: Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. arXiv preprint arXiv:2211.12588 (2022)

[26] Wu, Z., Han, C., Ding, Z., Weng, Z., Liu, Z., Yao, S., Yu, T., Kong, L.: Os-copilot: Towards generalist computer agents with self-improvement. arXiv preprint arXiv:2402.07456 (2024)

[27] Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al.: Least-to-most prompting enables complex reasoning in large language models. arXiv preprint arXiv:2205.10625 (2022)

[28] Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., Sabharwal, A.: Decomposed prompting: A modular approach for solving complex tasks. arXiv preprint arXiv:2210.02406 (2022)

[29] Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R.K.-W., Lim, E.-P.: Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (volume 1: Long Papers), pp. 2609–2634 (2023)

[30] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. Advances in neural information processing systems **36**, 8634–8652 (2023)

[31] Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al.: Inner monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:2207.05608 (2022)

[32] Li, G., Hammoud, H., Itani, H., Khizbullin, D., Ghanem, B.: Camel: Communicative agents for" mind" exploration of large language model society. Advances in neural information processing systems **36**, 51991–52008 (2023)

[33] Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., *et al.*: Autogen: Enabling next-gen llm applications via multi-agent

conversations. In: First Conference on Language Modeling (2024)

[34] Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S.K.S., Lin, Z., *et al.*: Metagpt: Meta programming for a multi-agent collaborative framework. In: The Twelfth International Conference on Learning Representations (2023)

[35] Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., *et al.*: Chatdev: Communicative agents for software development. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (volume 1: Long Papers), pp. 15174–15186 (2024)

[36] Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., Stone, P.: Llm+ p: Empowering large language models with optimal planning proficiency. arXiv preprint arXiv:2304.11477 (2023)

[37] Guan, L., Valmeekam, K., Sreedharan, S., Kambhampati, S.: Leveraging pretrained large language models to construct and utilize world models for model-based task planning. Advances in Neural Information Processing Systems **36**, 79081–79094 (2023)

[38] Hao, S., Gu, Y., Ma, H., Hong, J., Wang, Z., Wang, D., Hu, Z.: Reasoning with language model is planning with world model. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 8154–8173 (2023)

[39] Pan, L., Albalak, A., Wang, X., Wang, W.: Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In: Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 3806–3824 (2023)

[40] Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., Haq, S., Sharma, A., Joshi, T.T., Moazam, H., et al.: Dspy: Compiling declarative language model calls into self-improving pipelines. arXiv preprint arXiv:2310.03714 (2023)

[41] Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., et al.: Agentbench: Evaluating llms as agents. arXiv preprint arXiv:2308.03688 (2023)

[42] Mialon, G., Fourrier, C., Wolf, T., LeCun, Y., Scialom, T.: Gaia: a benchmark for general ai assistants. In: The Twelfth International Conference on Learning Representations (2023)

[43] Li, D., Murr, L.: Humaneval on latest gpt models–2024. arXiv preprint arXiv:2402.14852 (2024)

[44] Yao, S., Shinn, N., Razavi, P., Narasimhan, K.: Tau-bench: A benchmark for tool-agent-user interaction in real-world domains. arXiv preprint arXiv:2406.12045 (2024)

[45] Wang, X., Wang, Z., Liu, J., Chen, Y., Yuan, L., Peng, H., Ji, H.: Mint: Evaluating llms in multi-turn interaction with tools and language feedback. arXiv preprint arXiv:2309.10691 (2023)

[46] Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., Kambhampati, S.: Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. Advances in Neural Information Processing Systems **36**, 38975–38987 (2023)

[47] Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., et al.: Toolllm: Facilitating large language models to master 16000+ real-world apis. arXiv preprint arXiv:2307.16789 (2023)

[48] He, G., Demartini, G., Gadiraju, U.: Plan-then-execute: An empirical study of user trust and team performance when using llm agents as a daily assistant. In: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, pp. 1–22 (2025)

[49] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. Advances in neural information processing systems **36**, 11809–11822 (2023)