

Agent Reasoning Tools (ARTs): A Tool Definition Approach for Empower LLM-based Agent Systems

Jie Tao
Fairfield University
jtao@fairfield.edu

Lina Zhou
The University of North Carolina
at Charlotte
lzhou8@charlotte.edu

Abstract

The emergence of LLM-based agentic systems is transforming human-technology interaction by enabling proactive collaboration. However, LLMs often rely on external tools due to their lack direct interaction with environments or access to up-to-date information. This makes effective tool definition and management essential, yet such efforts are challenged by rigidity, overhead, and complexity of logic specification. Current methods often focus on external capabilities, overlooking the enhancement of an agent's internal reasoning. This research introduces Agent Reasoning Tools (ARTs) to address these challenges. ARTs are designed to reduce rigidity and overhead while enabling flexible, human-understandable logic definitions that enhance human-AI collaboration. Evaluating ARTs on aspect term extraction shows highly competitive performance. They represent a significant step toward more flexible, transparent, and user-friendly agentic systems.

Keywords: LLM, AI agents, tool use, reasoning, agentic systems.

1. Introduction

The emergence of Large Language Model (LLM)-based agentic systems reshapes how humans interact with technology and collaborate in the digital space. Importantly, LLMs have moved beyond passive information retrieval and content generation to become proactive, autonomous entities capable of interacting with the environment, making decisions and taking actions to achieve specific goals (Masterman et al., 2024; Xi et al., 2025). LLM-based agentic systems hold tremendous potential to augment human abilities. By serving as proactive collaborators in digital environments, they can help unlock the full capabilities of both LLMs and human-LLM collaboration.

Although agents have long been employed for task execution in distributed environments, a key recent advancement is leveraging LLMs, particularly their reasoning and planning abilities, to improve coordination and decision-making (Amant & Wood, 2005). However, LLMs do not directly interact with their environments or access up-to-date information. Instead, they often rely on tools such as web search, external databases access, and code execution. As a result, the effectiveness of such agents are directly impacted by the tool reliability and availability (Patil et al., 2024; Schick et al., 2023).

Effectively defining (specifying purpose, scope, functionality) and managing (implementing, maintaining, optimizing) tools in LLM-based agentic systems is essential yet presents significant challenges. (Masterman et al., 2024; Qin et al., 2024). First, agentic systems require tools to align with high-level objectives (e.g., decision-making). Consequently, defining tools that support these goals while remaining flexible for diverse tasks is difficult. Second, the integration of LLMs with these tools requires bridging different interfaces and data formats, posing a technical challenge. Third, managing tools to provide users with control and transparency is essential, especially in agentic systems where autonomy can obscure decision-making. This is because lack of transparency can reduce user trust, while insufficient control can lead to unintended outcomes (Liao & Wortman Vaughan, 2024; L. Zhou et al., 2023). However, existing tool definition methods do not adequately address these challenges and show several limitations. These include the rigidity of Application Programming Interfaces (APIs) and schemas (Google AI, n.d.; OpenAI, n.d.), significant implementation overhead, and the difficulty of specifying qualitative logic. Critically, current methods focus on granting external capabilities to LLM-based agents, while largely overlooking enhancing their internal reasoning capabilities.

This research introduces Agent Reasoning Tools (ARTs) to address the above-mentioned challenges and

limitations. Following a design science research (DSR) approach (Gregor & Hevner, 2013), this work introduces ARTs as a novel design artifact. ARTs provide a mechanism for defining agent capabilities through a formal function signature and a detailed natural language docstring. The underlying LLM directly interprets this specification for execution, primarily to enhance internal reasoning. ARTs are designed to overcome the rigidity and overhead of traditional schema-based or code-centric tool definitions. They also enhance flexibility in defining complex or internal reasoning logic, including implicit goals or qualitative judgments that the LLM interprets for execution. Moreover, the human-readable definitions help address transparency gaps, which are essential for fostering human-AI collaboration and building trust. These characteristics, in turn, facilitate more autonomous, semantically guided orchestration of the reasoning processes. We evaluate the effectiveness of ARTs with the task of Aspect Term Extraction (ATE), and our experiment results demonstrate their highly competitive performance. Therefore, ARTs represent a significant step toward more flexible, transparent, and user-friendly agentic systems.

2. Related work

2.1 Need for tool use in agentic systems

One crucial aspect of AI Agents is their ability to take actions, which in turn alter the world state. However, LLMs do not directly interact with their environments or access up-to-date information, so tool-enabled frameworks and environments become key. Tools are essentially functions that extend the agent's capabilities by allowing it to perform specific actions or fulfill a clear objective (Mavroudis, 2024). The industry is shifting toward enhancing AI Agent capabilities through tool access rather than simply improving model power.

The power and flexibility of LLM-based AI agents depend heavily on tool access. Effective tools should enhance the capabilities of LLMs. For example, providing an LLM with a calculator tool for arithmetic tasks yields more accurate results than depending on its native capabilities. In addition, the internal knowledge of LLMs, being confined to their training data, typically lags significantly, making them prone to hallucination when handling up-to-date information such as interest rates without a search tool. Therefore, integrating LLMs with external tools is essential to extend their capabilities and enhance the overall performance (Shen et al., 2024).

2.2 Tool use definitions and processes

Effective tool use requires clear definitions or descriptions of how they work. Traditionally, a tool use is treated as a function calling and its description consists of the following elements: 1) A textual description of what the function does; 2) a callable (something to perform an action); and 3) arguments with typings. An optional component of the definition is output with typings (Hugging Face, n.d.).

Once a tool is defined, LLMs can use it to complete relevant tasks. However, since LLMs inherently process and generate only text, they lack native tool-calling capabilities. Therefore, providing tools to an agent requires informing the LLM about their existence and instructing it to generate text-based invocations when needed. Within LLM-based agents, this tool use is typically described through a system prompt in natural language and activated through the agent's internal reasoning process (Kumar, 2024). The system prompt contains elements of tool definitions such as function calling. Yet one notable difference in tool use from function calling is that it is fully descriptive and flexible. Here is a simple illustration of the tool definition: `def calculator(a: float, b: float) -> float: "Adding two floats"`. When an LLM receives it, the model will recognize it as a tool and will know what it needs to pass as inputs and what to expect from the output. Excluding the outputs with typings in the definition will allow the agent to determine how to implement the tool (or directly generate tool calls based on its observations) during tool use, supporting flexibility, reusability while keeping the world state tracking implicit (Lu et al., 2025).

Despite various tool definitions from major providers of agent tools, including OpenAI's Swarm Operator, LangChain agents, Anthropic Functions, Microsoft's AutoGen, Google Gemini plus Bard Extensions, and Amazon Bedrock's AI Agent framework, existing work largely relies on closed-source LLM APIs to interpret or learn to solve complex tasks with LLM-based agents (Jiang et al., 2024). As a result, they remain rigid and lack sufficient flexibility required to interact with different types of LLMs.

2.3 Agentic workflows and multi-agent collaboration

Moving beyond single tool invocations, agents can engage in multi-step processes or collaborate within workflows. Workflow typically refers to "a set of subtasks with execution dependencies" (Zhang et al., 2025). Agentic workflows mark a significant paradigm shift in deploying LLMs. In contrast with non-agentic workflow, where LLMs rely on ongoing specific

feedback from users to enhance or refine its output through back-and-forth user-agent interaction, agentic workflow involves LLM agents that proactively engage in a series of questions and explorations on its own to enhance the outcome through a self-guided cycle of feedback and refinement despite that it also begins with the user initiating the first query. Agentic workflow not only offers flexibility but also has the potential to deliver better outcomes (Andreoni et al., 2024).

Four foundational pillars underpin the agentic workflow: reflection, tools, planning, and multi-agent collaboration (DeepLearning.AI., n.d.). Tool use extends LLMs' capabilities beyond language processing, as discussed earlier. Reflection is a crucial aspect of agentic reasoning, allowing systems to analyze and improve their outputs iteratively. Planning involves formulating a sequence of actions to achieve specific goals. Multi-agent collaboration is pivotal for systems to perform towards common objectives that surpass the capabilities of individual agents by sharing tasks and insights (Qiao et al., 2025).

Evaluator-Optimizer is one of the agentic workflow patterns in which two types of LLMs with two distinct roles involved in a continuous feedback loop to enhance the quality of generated outputs, with one role creating initial responses based on the given prompt while the other providing evaluation and feedback to refine and optimize the outputs iteratively. This is in direct contrast to sequential workflow. Nevertheless, the issue with Evaluator-Optimizer workflow may lie in determining when to stop the feedback loop and in instructing the model to stop editing the output once an established set of criteria is met, which also requires clear evaluation criteria (Briva-Iglesias, 2025).

Complex tasks typically involve complex reasoning. Constructing complex workflow raises many challenges, including performance, portability, and productivity challenges (Ben-Nun et al., 2020). Higher-level interfaces and modularity are among the performance challenges, and containers (easy to deploy for each target platform) illustrate portability challenges, and real-time feedback (automating processing to avoid human latencies) and common data representations manifests performance challenges.

3. The Agent Reasoning Tool (ART) approach

Autonomous agents, particularly those powered by LLMs, increasingly rely on external tools to interact with environments, access knowledge, and perform complex actions (Masterman et al., 2024). Conventional approaches typically involve exposing tools via strictly defined API schemas mapped to executable functions (e.g., standard function calling mechanisms) or defining

tools directly via imperative code (Schick et al., 2023). While functional, these methods can be rigid, demand significant implementation effort, and may limit the agent's capacity to adapt tool usage flexibly based on context, as discussed in Section 2. Moreover, defining complex, nuanced behaviors solely through code or structured schemas can be challenging.

To address these constraints and foster greater agent adaptability and expressiveness in tool definition, this paper introduces the Agent Reasoning Tool (ART) approach, a novel design artifact. ARTs contribute a nascent design science theory (Gregor and Hevner, 2013), as it provides a new method, architecture, and a set of operational principles for tool definition that fosters greater agent adaptability and expressiveness. Conceptually, an ART defines a tool's function using a combination of a formal signature and a detailed natural language description (docstring), which can be interpreted directly by the agent's underlying LLM. It differs from complex prompt engineering by defining a reusable tool structure rather than instance-specific instructions, and from standard function calling by relying on LLM interpretation of the docstring for core logic execution rather than simply mapping to pre-existing code through structured descriptions. This section details the definition, structure, execution and orchestration of ARTs.

3.1 Definition and structure

From an Information Systems perspective, any functional component can be understood via its inputs, processing, and outputs. Adopting a structure similar to Python-style docstrings is advantageous because their prevalence in LLM training data can potentially lead to better interpretation by the model, and easy-to-read by human users as well. Accordingly, the proposed ARTs blend formal interface specifications with rich, natural language descriptions. Specifically, an ART formally comprises:

- **Function Signature:** Defines the tool's name, required input parameters (with types), and expected output type(s). This establishes the structural contract (the "shape" of input/output).
- **Natural Language Docstring:** As the functional specification, it details purpose and logic of the tool:
 - **Input Semantics and Constraints:** Elaborates on the meaning and expected characteristics of inputs beyond simple types.
 - **Processing Logic Specification:** This segment describes the intended transformation or reasoning the tool

performs. Crucially, for ARTs, this often involves a high-level description of the tool's goal and behavior, rather than explicit, step-by-step imperative code provided by the developer for the core logic. The underlying LLM is responsible for interpreting this natural language specification, along with the defined inputs and expected outputs, to infer and execute the necessary processing steps. Figure 1 presents an example ART definition (`single_eval`) where the docstring provides such a high-level specification. Moreover, the flexibility of this natural language definition allows for specifying

composite operations. This means an ART can be defined to internally invoke other tools, including other ARTs or traditional coded functions, as part of its described logic. While the concise docstring in Figure 1 focuses on the high-level task, this composition capability is a key feature of ARTs, facilitating modular design. This will be discussed further in Section 3.2 and demonstrated in Section 4.2.

- **Output Semantics and Structure:** Details the meaning, structure, and content requirements of outputs, guiding the LLM to generate responses aligned with the tool's purpose, thus regulating the output.

```
# --- Agent Reasoning Tool (ART) Definition: single_eval ---

# ANNOTATION 1: Formal interface defining tool name, typed inputs, & outputs.
# Function Signature:
def single_eval(review_text: str,
                extracted_aspects: list,
                actual_aspects: list) -> tuple[dict, str]: # Output types implied by docstring
    # ANNOTATION 2: High-level semantic description of purpose, inputs, and expected outputs.
    # The LLM infers the necessary processing logic from this specification.
    # Natural Language Docstring (LLM-Interpretable Functional Specification):
    """
    Given review_text, compare extracted_aspects with actual_aspects to identify any discrepancies.
    INPUT:
    - review_text (str)
    - extracted_aspects (list): list of aspects extracted from review_text
    - actual_aspects (list): list of human annotated aspects from review_text
    OUTPUT:
    - revised_aspect_dict (dict):
      - keys: extracted_aspects
      - values: reasons to add, remove, or retain certain aspect
    - feedback (str): feedback to improve the extraction process/results. Return "" if
      the extracted_aspects is the same as actual_aspects.
    """
    # For an ART, the LLM's interpretation of the docstring above guides its execution.
    # No explicit imperative code is typically written here by the developer for the core logic.
```

Figure 1. Example ART Definition.

Figure 1 illustrates ART definition (`single_eval`) including the docstring's processing logic. For instance, a directive to 'generate constructive feedback highlighting discrepancies between extracted and ground truth aspects, focusing on actionable advice for the Decision Component' – relies on the LLM to interpret 'constructive' and 'actionable' and formulate the appropriate nuanced feedback without explicitly listing every rule for feedback generation. This demonstrates the specification of implicit logic. By combining a formal signature with a detailed natural language docstring that specifies input semantics, processing logic (both explicit and implicit), and detailed output

requirements, the ART provides a rich, interpretable definition that LLM-based agents can leverage for execution while providing structure and regulation to the generated output.

3.2 Execution and orchestration

The execution of an individual ART represents a distinct approach compared to conventional tool invocation. Unlike tools executing pre-compiled code or calling structured APIs, an ART is "executed" when the agent's LLM processes its full definition - signature and detailed natural language docstring - along with input values. The LLM interprets this to implement the

described logic and generate output, leveraging its reasoning capabilities (Jiang et al., 2024; Wei et al., 2022). ARTs complement conventional methods, which use coded logic or schemas for deterministic operations (Google AI, n.d.; OpenAI, n.d.), by describing what a tool should do in natural language, leaving the LLM to infer the how for its internal reasoning steps. This design inherently promotes flexibility and semantic expressiveness, allowing for the natural specification of complex or qualitative logic.

Most importantly, the ART mechanism extends beyond single tool execution to facilitate the autonomous orchestration of multiple tools by the agent. Given a task, instructions, and available ART definitions, the LLM first interprets each ART's semantics and then, assessing the context, autonomously plans and selects the next ART or sequence of ARTs. This dynamic planning, driven by the LLM's understanding of the rich semantic content within ART definitions, distinguishes the ART approach from workflows with hardcoded sequences or rigid rules.

This interpretative execution model of an ART unlocks its several key properties. Compositionality allows complex ARTs to be built modularly. For instance, an ART designed for a nuanced evaluation task could internally invoke another ART for data parsing, or even a traditional coded function. This highlights the complementary nature of ARTs: an agent might employ an ART for qualitative assessment based on its natural language definition, and then use an explicitly coded function (e.g., `calc_metrics`) for precise quantitative calculations. Furthermore, because the ART's logic resides in its human-readable docstring, the approach offers inherent explainability, making agent actions more transparent. The detailed specifications within ARTs also contribute to output regulation, guiding the LLM towards more consistent and predictable results. While sharing goals with agent frameworks that incorporate reasoning steps (e.g., ReAct (Yao et al., 2023)), this specific orchestration mechanism emphasizes the LLM's direct reasoning based on the provided task context and the rich semantic content of the ART definitions themselves. Section 4 will provide a concrete use case for this approach.

4. Agentic workflow with ARTs

4.1 Experiment setup and use case

In this experiment, we used Google's Gemma 3 27B model for all agents and baseline(s). We selected this state-of-the-art open model within its parameter class due to its strong performance on various language understanding and generation benchmarks, while

maintaining computational accessibility for research compared to significantly larger frontier models (Gemma Team et al., 2025). Its open nature also enhances transparency and reproducibility. We set the model's temperature to 1 to maximize output diversity, allowing us to rigorously test our hypothesis that the structured ART definitions can effectively regulate responses and guide the agent towards deterministic task completion even with high randomness. While this temperature might not optimize raw F1 scores, it serves our experimental goal of evaluating ARTs' regularization. Our primary prompting strategy was zero-shot with one specific exception: the Decision Component's autonomous ART selection and orchestration prompt employed one-shot learning, providing a single illustrative tool-use sequence to aid its planning process.

We employed two baselines in this experiment. The first involves fine-tuning transformer models, a widely adopted technique in recent studies on this dataset. The fine-tuned models serve as the state-of-the-art benchmark, addressing the performance question of whether the ART framework, without any task-specific fine-tuning, can achieve performance comparable to specialized, high-effort methods. The second baseline (Baseline-PromptChain) consists of directly prompting the chosen LLM to perform ATE, thus bypassing our iterative workflow or the use of specific ARTs. This baseline provides a crucial direct comparison that isolates the specific contribution of the structured ART workflow by comparing an LLM enhanced with the workflow against the same LLM in a simple prompting setup. For a fair comparison, Baseline 2 also utilized the Gemma 3 27B model with Temperature=1 and a zero-shot approach for the core task. Additionally, to assess the impact of the iterative refinement process itself, we evaluated a "Single-Agent Workflow." In this configuration, only the Decision Component operates, performing its aspect term extraction in a single pass using its suite of ARTs, without the iterative feedback loop involving the Evaluation Component. Another configuration is a "Multi-Agent Workflow", in which we calculated the F1-score using a traditional coded function (`calc_metric`) executed by the Evaluation Agent. The iterative workflow terminates when either the F1-score reached or exceeded a predetermined threshold (e.g., 0.85), indicating satisfactory performance, or the number of iterations reached a maximum limit (e.g., 2), triggering a fallback policy (e.g., selecting the output from the iteration with the highest F1-score).

We evaluated our approach on the SemEval 2014 Task 4 (Subtask 1) dataset (Pontiki et al., 2014), a standard benchmark widely used in recent Aspect Term Extraction (ATE) research, which allows for direct

comparison with existing work. For evaluation, we used the “restaurant” reviews and the test split, which contains 800 reviews. These reviews are labeled with five aspects: food, service, price, ambience, and anecdotal/miscellaneous. We selected weighted-average F1-score as the evaluation metrics, prevalent in SemEval ATE evaluations. This metric is suitable for ATE’s multi-label nature, as a single review may contain multiple aspect terms. To evaluate the agents while staying aware of the context window size, we performed repeated sampling with replacement for 40 times, with a sample size of 150 reviews each time.

4.2 ATE workflow: design, roles, and process

To demonstrate ARTs in a practical setting, we implemented an iterative workflow for ATE. This workflow draws inspiration from an Evaluator-Optimizer iterative refinement pattern and involves two

principal functional units: a Decision Component and an Evaluation Component, each utilizing ARTs for its core functions. The architecture, the involved components, and their respective toolsets are illustrated in the component diagram in Figure 2.

- Decision Component:** This component is responsible for processing the input review text and generating or revising a list of candidate aspect terms. It achieves this by autonomously orchestrating a suite of granular ARTs (`add_aspect`, `remove_aspect`, `check_aspects`, `end_extraction`). This autonomous orchestration within the component relies on its interpretation of the ARTs’ natural language docstrings and any received feedback, showcasing the agentic reasoning capabilities ARTs can enable within a defined functional role.

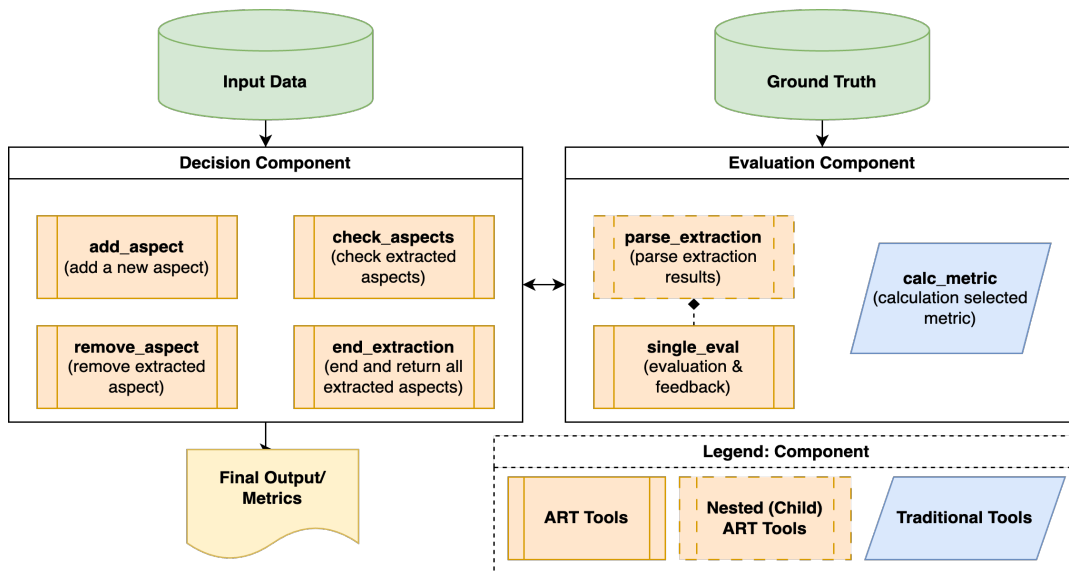


Figure 2. ARTs in the ATE Workflow.

- Evaluation Component:** This component assesses the list of aspect terms identified by the Decision Component for each review and provides evaluative feedback. It employs a hybrid toolset:
 - Its primary `single_eval` ART defines the qualitative assessment and feedback generation logic in natural language for a single review instance.
 - This `single_eval` ART internally invokes a `parse_extraction` ART to convert the Decision Component’s output into a structured format, demonstrating the compositionality of ARTs.
 - A traditional coded Python function, `calc_metric`, is used for quantitative F1-score calculation, illustrating the complementary nature of ARTs with existing coded tools.
- The operational flow of this iterative process is depicted in the flowchart in Figure 3. The workflow progresses through the following steps:
- Initial Extraction:** The Decision Component processes the input review text by autonomously calling its suite of granular

ARTs to generate an initial list of aspect terms, finalized via its `end_extraction` ART.

- **Parse & Eval:** Upon receiving the list, the Evaluation Component’s `single_eval` ART first invokes its internal `parse_extraction` ART. It then performs the qualitative comparison and generates feedback data. Subsequently, the `calc_metric` coded function computes the F1-score.
- **Decision Point 1 (Satisfaction Check):** If the F1-score meets the target threshold, then the workflow concludes.
- **Decision Point 2 (Termination Check):** If the F1 falls below the target threshold, the iteration count reaches its maximum limit, the workflow invokes the fallback policy and terminates.
- **Generate Feedback:** If termination conditions are not met, the Evaluation Component’s `single_eval` ART formulates natural language feedback based on its qualitative comparison results.
- **Revise Extraction:** The Decision Component receives this feedback, interprets it, and initiates a revision pass by again orchestrating calls to its granular ARTs before finalizing with `end_extraction`. The workflow then returns to Step 2 (**Parse & Eval**).

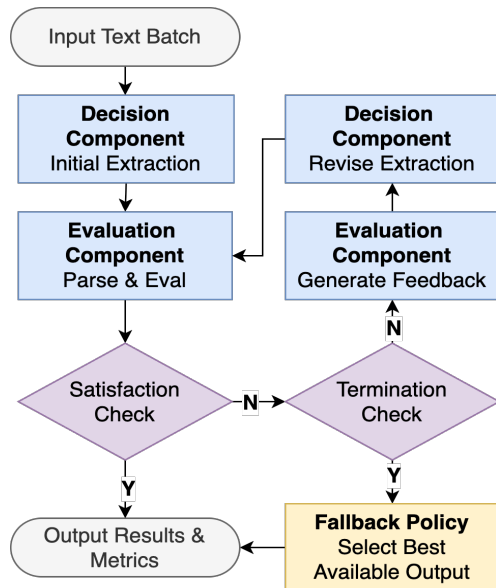


Figure 3. Agentic Workflow Design.

This detailed description illustrates the structure and operation of our ATE workflow, highlighting that ARTs are integral to the functionality of each component and the overall iterative process, thereby

demonstrating the practical application of the ART mechanism.

5. Results and discussion

5.1 Results

The evaluation involved comparing three configurations with ART: Baseline 1, which leveraged benchmark results from fine-tuned models in the literature (Scaria et al., 2024; Yan et al., 2022); Baseline 2, which utilized direct LLM prompting; Single-Agent Workflow (SingleAgent in Figure 4), which employed the Decision Component with ARTs in a single pass, as well as Multi-Agent Workflow (MultiAgent in Figure 4), which is the evaluation-optimization interactions between the Decision and Evaluation Component. The detailed results of this comparison are presented in Table 1.

Table 1. Performance Comparisons

(a) Descriptive Statistics of Performances			
	MEAN	STDEV	MEDIAN
Baseline1	0.8687	0.0407	0.8707
Baseline2	0.7109	0.1071	0.7406
SingleAgent	0.8044	0.0790	0.8082
MultiAgent	0.9103	0.0623	0.9290
(b) Statistical Significance of Pairwise Comparison			
	Baseline 1	Baseline 2	SingleAgent
Baseline 1	-		
Baseline 2	p<.1	-	
SingleAgent	p<.01	p<.001	-
MultiAgent	p<.01	p<.001	p<.001

As shown in Table 1 (a), the ART-based workflows demonstrate a clear and significant performance advantage. The direct prompting of Baseline 2 yielded a mean F1-score of 0.7109 with considerable variance (SD = 0.1071). In contrast, the Single-Agent Workflow achieved a much higher and more stable F1-score of 0.8044 (SD = 0.079). The Multi-Agent Workflow achieved the best performance, with a mean F1-score of 0.9103 and the lowest variance (SD = 0.0623). Moreover, the performance improvements driven by the ART framework are statistically significant, highlighting its superior accuracy and stability. As noted in Table 1, the introduction of the structured, single-pass ART workflow provided a significant performance gain over the unstructured prompting of Baseline 2. Furthermore, the iterative, multi-agent refinement process led to a second most significant increase in F1-score over the single-pass workflow (p < 0.001 for both comparisons). These results confirm that the primary

source of these gains is the ART-driven agentic structure, not just the LLM itself.

When contextualized against Baseline 1, the Multi-Agent Workflow achieves a performance (mean =0.91, median=0.93) that competes with and potentially surpasses the upper range (typically ~0.85 to 0.92) reported for strong traditional fine-tuned models. This finding is particularly noteworthy, suggesting that our agentic workflow using ARTs can achieve state-of-the-art results on this benchmark without the need for task-specific model fine-tuning. While the Single-Agent Workflow underperforms the multi-agent approach, it still surpasses the direct prompting baseline and the performance levels of some earlier or less optimized fine-tuned models reported in the literature.

5.2 Research implications

The superior performance of the ART-based workflow, alongside the novel design of ARTs themselves, presents several significant implications for future AI agent development. Firstly, ARTs offer a distinct advantage in defining agent capabilities, particularly for nuanced internal reasoning. By leveraging direct LLM interpretation of detailed natural language docstrings combined with formal signatures, this approach demonstrated notable flexibility. The specification of complex qualitative logic for the `single_eval` ART, combined with the Decision Component's autonomous orchestration of its granular ARTs, provides a powerful method for creating agents with sophisticated, semantically-driven behaviors that complement traditional, often externally-focused, tool-use paradigms. The ATE workflow showcased ARTs' capacity to strengthen internal reasoning. For instance, the `single_eval` ART empowered the Evaluation Component to perform nuanced qualitative comparisons and generate context-specific feedback, tasks that are inherently about internal cognitive processing rather than external data retrieval. Furthermore, the Decision Component's ability to dynamically manage its own suite of extraction-related ARTs in response to this feedback illustrates an LLM-driven internal planning and execution cycle, a core pillar of the ART approach.

Second, following the DSR Knowledge Contribution Framework (Gregor and Hevner, 2013), which classifies research contributions based on the maturity of the problem and solution domains, this study constitutes an "Improvement" contribution by proposing a novel solution (ARTs) to a well-understood problem (the need for effective and flexible tool definition for AI agents). The ART artifact offers a new method and operational principles that demonstrably outperform existing solutions, such as rigid APIs and code-centric definitions. More septically, this positions our work as a

Level 2 DSR contribution—a nascent design theory—that provides generalizable knowledge for designing and building more capable agentic systems.

Third, this research highlights a promising pathway to achieving competitive performance in complex tasks without necessitating task-specific model fine-tuning. The high F1 scores attained by the ART-driven ATE workflow underscore the potential of structured iterative refinement, guided by rich, ART-defined feedback (such as that from `single_eval`), to match and even exceed benchmarks set by fine-tuned models. This has practical implications for reducing the development overhead and data dependency often associated with specializing models for niche applications.

Last, the ART mechanism inherently fosters more modular, accessible, and explainable AI development practices. The modularity, demonstrated by the `single_eval` ART composing the `parse_extraction` ART and its seamless integration with the coded `calc_metric` function, points towards building robust and versatile agent toolkits. The natural language basis for defining complex logic may also enhance accessibility, potentially lower technical barriers and allow broader expertise to contribute to AI behavior specification. Moreover, because the human-readable docstring also serves as the functional specification, ARTs offer inherent explainability. This transparency in how agent tools are defined and thus how components reason can facilitate clearer human-AI understanding, streamline debugging, and foster greater trust in agentic systems.

Despite these advantages, effective practical deployment necessitates a careful consideration of the associated trade-offs. While the ART-based iterative workflow yielded high accuracy, its multi-step nature can entail greater inference latency and computational costs compared to single-pass or highly optimized fine-tuned models. The optimal approach will depend on specific application constraints, balancing performance needs with development efficiency and interpretability requirements. Future research should focus on validating ARTs across a wider array of tasks and LLMs, developing robust tooling for ART creation and management, and further exploring the capabilities and limits of LLM interpretation in executing these natural language-defined tools.

6. Concluding remarks

This research introduced ARTs as a novel mechanism to enhance AI agents with adaptability, transparency, and ease-of-use, crucial for effective human-AI co-creation and trust-building in digital collaboration. ARTs primarily enhance the internal

reasoning of LLM-based agents, distinct from traditional tool definitions that focus on external capabilities. Our experiments show that ARTs enable structured interaction between functional components within an iterative workflow while enhancing human agency through interpretable feedback and traceable actions. Practically, ARTs offer an accessible way to define complex AI behaviors, fostering hybrid teams where AI augments human capabilities to enhance collective intelligence. They represent a promising pathway toward more human-centered AI collaboration.

This study has several limitations. First, the evaluation was focused on a single task and LLM, limiting the generalizability of our findings. Second, manually creating ART definitions may not represent the most effective design. Third, the orchestration prompting techniques require further detail for full reproducibility. Fourth, we did not measure inference cost/latency in iterative agentic workflows compared to single-pass or simpler methods, preventing a complete analysis of trade-offs for practical deployment. Finally, future work should validate ARTs' performances across diverse human-AI collaboration scenarios.

7. References

- Acharya, D. B., Kuppan, K., & Divya, B. (2025). Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey. *IEEE Access*, 13, 18912–18936. <https://doi.org/10.1109/ACCESS.2025.3532853>
- Amant, R. St., & Wood, A. B. (2005). Tool use for autonomous agents. *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*, 184–189.
- Andreoni, M., Lunardi, W. T., Lawton, G., & Thakkar, S. (2024). Enhancing Autonomous System Security and Resilience With Generative AI: A Comprehensive Survey. *IEEE Access*, 12, 109470–109493. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2024.3439363>
- Ben-Nun, T., Gamblin, T., Hollman, D. S., Krishnan, H., & Newburn, C. J. (2020). Workflows are the New Applications: Challenges in Performance, Portability, and Productivity. *2020 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, 57–69. <https://doi.org/10.1109/P3HPC51967.2020.00011>
- Briva-Iglesias, V. (2025). Are AI agents the new machine translation frontier? Challenges and opportunities of single- and multi-agent systems for multilingual digital communication (No. arXiv:2504.12891). *arXiv*. <https://doi.org/10.48550/arXiv.2504.12891>
- Building Effective AI Agents. (n.d.). Retrieved May 27, 2025, from <https://www.anthropic.com/engineering/building-effective-agents>
- Four AI Agent Strategies That Improve GPT-4 and GPT-3.5 Performance. (n.d.). Retrieved May 27, 2025, from <https://www.deeplearning.ai/the-batch/how-agents-can-improve-llm-performance/?ref=dl-staging-website.ghost.io>
- Function Calling with the Gemini API | Google AI. (n.d.). Retrieved May 27, 2025, from <https://ai.google.dev/gemini-api/docs/function-calling>
- Function calling | OpenAI. (n.d.). Retrieved May 27, 2025, from <https://platform.openai.com>
- Gemma Team, Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., ... Hussenot, L. (2025). Gemma 3 Technical Report (No. arXiv:2503.19786). *arXiv*. <https://doi.org/10.48550/arXiv.2503.19786>
- Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *MIS Quarterly*, 37(2), 337–355.
- Introduction to smolagents | Hugging Face. (n.d.). Retrieved May 27, 2025, from <https://huggingface.co/learn/agents-course/en/unit2/smolagents/introduction>
- Jiang, J., Zhou, K., Zhao, W. X., Song, Y., Zhu, C., Zhu, H., & Wen, J.-R. (2024). KG-Agent: An Efficient Autonomous Agent Framework for Complex Reasoning over Knowledge Graph (No. arXiv:2402.11163). *arXiv*. <https://doi.org/10.48550/arXiv.2402.11163>
- Kumar, S. (2024, August 25). Agent & Tools—Basic Code using LangChain. *Medium*. <https://medium.com/@shravankoninti/agent-tools-basic-code-using-langchain-50e13eb07d92>
- Liao, Q. V., & Wortman Vaughan, J. (2024). AI Transparency in the Age of LLMs: A Human-Centered Research Roadmap. *Harvard Data Science Review, Special Issue 5*. <https://doi.org/10.1162/99608f92.8036d03b>
- Lu, J., Holleis, T., Zhang, Y., Aumayer, B., Nan, F., Bai, F., Ma, S., Ma, S., Li, M., Yin, G., Wang, Z., & Pang, R. (2025). ToolSandbox: A Stateful, Conversational, Interactive Evaluation Benchmark for LLM Tool Use Capabilities (No. arXiv:2408.04682). *arXiv*. <https://doi.org/10.48550/arXiv.2408.04682>
- Masterman, T., Besen, S., Sawtell, M., & Chao, A. (2024). The Landscape of Emerging AI Agent Architectures for Reasoning, Planning, and Tool Calling: A Survey (No. arXiv:2404.11584). *arXiv*. <https://doi.org/10.48550/arXiv.2404.11584>
- Mavroudis, V. (2024). *LangChain v0.3*. <https://doi.org/10.20944/preprints202411.0566.v1>
- Patil, S. G., Zhang, T., Wang, X., & Gonzalez, J. E. (2024). Gorilla: Large Language Model Connected with Massive APIs. *Advances in Neural Information Processing Systems*, 37, 126544–126565.
- Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., & Manandhar, S. (2014). SemEval-2014 Task 4: Aspect Based Sentiment Analysis. *8th International Workshop on Semantic Evaluation, SemEval 2014 - Co-Located with the 25th International Conference on Computational Linguistics, COLING 2014, Proceedings*, 27–35. <https://doi.org/10.3115/V1/S14-2004>
- Qiao, S., Fang, R., Qiu, Z., Wang, X., Zhang, N., Jiang, Y., Xie, P., Huang, F., & Chen, H. (2025). Benchmarking

- Agentic Workflow Generation (No. arXiv:2410.07869). arXiv. <https://doi.org/10.48550/arXiv.2410.07869>
- Qin, Y., Hu, S., Lin, Y., Chen, W., Ding, N., Cui, G., Zeng, Z., Zhou, X., Huang, Y., Xiao, C., Han, C., Fung, Y. R., Su, Y., Wang, H., Qian, C., Tian, R., Zhu, K., Liang, S., Shen, X., ... Sun, M. (2024). Tool Learning with Foundation Models. *ACM Comput. Surv.*, 57(4), 101:1-101:40. <https://doi.org/10.1145/3704435>
- Scaria, K., Gupta, H., Goyal, S., Sawant, S. A., Mishra, S., & Baral, C. (2024). InstructABSA: 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2024. 720–736. <https://doi.org/10.18653/v1/2024.naacl-short.63>
- Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. *Advances in Neural Information Processing Systems*, 36, 68539–68551.
- Shen, W., Li, C., Chen, H., Yan, M., Quan, X., Chen, H., Zhang, J., & Huang, F. (2024). Small LLMs Are Weak Tool Learners: A Multi-LLM Agent (No. arXiv:2401.07324). arXiv. <https://doi.org/10.48550/arXiv.2401.07324>
- Wang, Z., Mao, S., Wu, W., Ge, T., Wei, F., & Ji, H. (2024). Unleashing the Emergent Cognitive Synergy in Large Language Models: A Task-Solving Agent through Multi-Persona Self-Collaboration. In K. Duh, H. Gomez, & S. Bethard (Eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)* (pp. 257–279). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.naacl-long.15>
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 35(NeurIPS), 1–43.
- Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., ... Gui, T. (2025). The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2), 121101. <https://doi.org/10.1007/s11432-024-4222-0>
- Yan, H., Yi, B., Li, H., & Wu, D. (2022). Sentiment knowledge-induced neural network for aspect-level sentiment analysis. *Neural Computing and Applications*, 34(24), 22275–22286. <https://doi.org/10.1007/s00521-022-07698-0>
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. *International Conference on Learning Representations (ICLR)*. <https://par.nsf.gov/biblio/10451467-react-synergizing-reasoning-acting-language-models>
- Zhang, J., Xiang, J., Yu, Z., Teng, F., Chen, X., Chen, J., Zhuge, M., Cheng, X., Hong, S., Wang, J., Zheng, B., Liu, B., Luo, Y., & Wu, C. (2025). AFlow: Automating Agentic Workflow Generation (No. arXiv:2410.10762). arXiv. <https://doi.org/10.48550/arXiv.2410.10762>
- Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., & Wang, Y.-X. (2024, June 6). Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models. *Forty-first International Conference on Machine Learning*. <https://openreview.net/forum?id=njwv9BsGHF>
- Zhou, L., Rudin, C., Gombolay, M., Spohrer, J., Zhou, M., & Paul, S. (2023). From Artificial Intelligence (AI) to Intelligence Augmentation (IA): Design Principles, Potential Risks, and Emerging Issues. *AIS Transactions on Human-Computer Interaction*, 15(1), 111–135. <https://doi.org/10.17705/1thci.00185>