Comparing AI System Designs

Estimated time: **5** minutes

When designing AI systems, the correct approach depends on the task's complexity, adaptability needs, and operational requirements. Let's compare three paradigms: **single LLM features, structured workflows, and autonomous AI agents**.

After completing this reading, you will be able to:

- Distinguish among single LLM features, structured workflows, and autonomous AI agents.
- Identify appropriate use cases and limitations for each AI system design.
- Evaluate which paradigm best fits a given task's complexity and adaptability needs.
- Recognize real-world trends in hybrid AI system design.

Single LLM features: Simple, one-shot tasks

Imagine you want to quickly summarize a news article or translate a customer review. You simply input the text into a single LLM and instantly receive the summarized or translated output—no further steps required. At the most basic level, you can use LLMs for simple, single-turn tasks with no memory or context across calls.

Key characteristics

Single LLM features have the following key characteristics:

- **Stateless processing**: No retention of information or context across interactions.
- **Direct input-output flow**: Straightforward request-response mechanism.
- **Predefined tasks**: Suitable only for clearly defined, single-step actions.

Best uses

This paradigm is best for:

- Simple, well-defined tasks that require no memory or multi-step logic

Examples

This paradigm is appropriate for:

- Text summarization
- Sentiment classification
- Information extraction
- Translation

Advantages

Using single LLM features offers:

- **Speed and simplicity**: Fastest to build and run
- **Deterministic output**: Same input, same output
- **Low cost**: Minimal compute and orchestration overhead

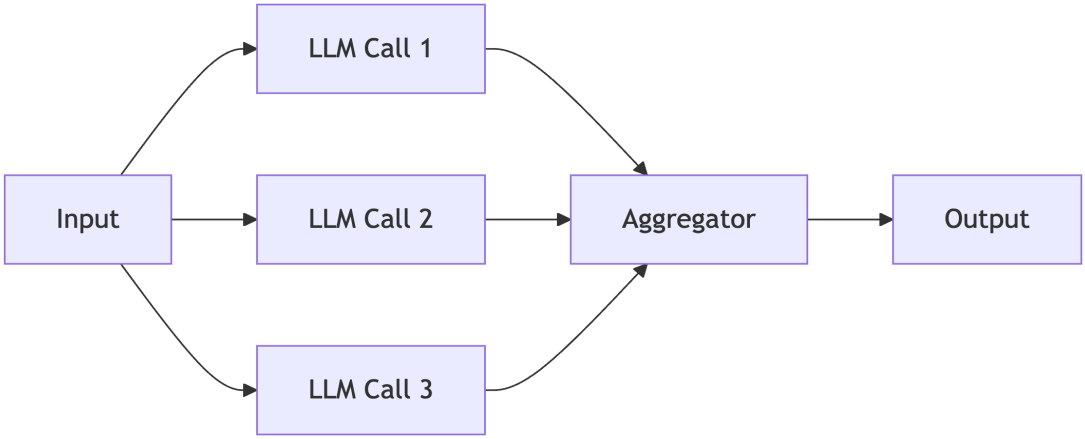
Limitations

When using a single LLM, you will likely encounter the following limitations:

- **No adaptability**: Cannot handle context or dynamic decision-making
- **No memory**: Each input is processed independently

Structured workflows: Multi-step, predictable processes

Structured workflows orchestrate LLM and tool calls through explicit, deterministic code paths. They're ideal for repetitive, multi-step, or compliance-heavy tasks. Consider processing insurance claims, where each document is scanned, information is extracted, validated, and stored. Each step must follow a precise, predictable order, making structured workflows ideal.



Key characteristics

Structured workflows have the following key characteristics:

- **Deterministic execution**: Inputs produce consistent outputs.
- **Explicit control flow**: All steps and decisions are predefined.
- **Predefined tool chains**: Tool use is fixed and transparent.

Best uses

Structured workflows work well for the following needs:

- Repetitive, multi-step tasks with clear logic and minimal ambiguity
- Regulatory or compliance-driven applications
- Scenarios requiring consistency, traceability, and auditability

Examples

You'll find structured workflows work well for the following scenarios:

- Document and data pipelines (Optical Character Recognition (OCR) → extraction → validation → storage)
- Batch report generation
- Financial and healthcare transaction processing

Advantages

Structured workflows offer the following advantages:

- **Predictable and reliable**: Easy to monitor, debug, and audit
- **Cost-efficient**: No unnecessary exploration
- **Compliance-ready**: Supports versioning, error handling, and audit trails

Limitations

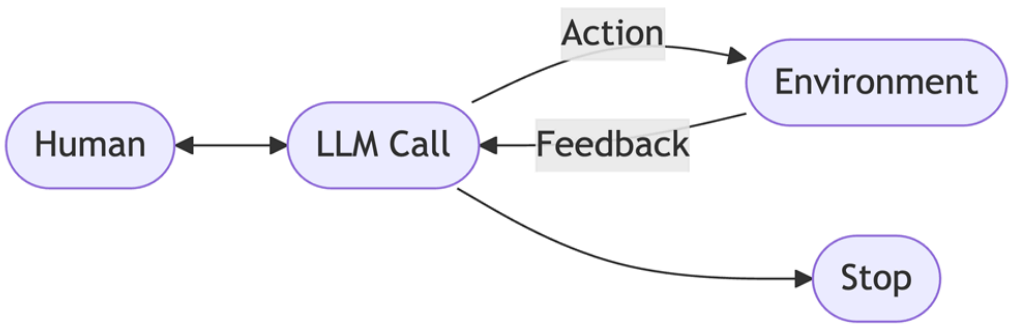
When using structured workflows, you might encounter the following limitations:

- **Rigidity**: Difficulty adapting to new or ambiguous scenarios
- **Development overhead**: The necessity to code each exception or variant

---

Autonomous agents: Flexible, context-aware reasoning

Autonomous agents allow LLMs to plan sequence actions and adapt as conditions change. Agents choose which tools to use and how to achieve their goals based on real-time context and feedback. Imagine an AI-driven virtual assistant helping a user plan a vacation. It dynamically gathers user preferences, researches destinations, suggests accommodations, and adapts recommendations based on feedback. This requires an autonomous agent capable of planning, context-awareness, and iterative improvement.



Core capabilities

Autonomous agents have the following core capabilities:

- **Dynamic planning**: Decomposes goals and adjusts steps as needed
- **Contextual awareness**: Remembers past steps and adapts to user and environment feedback
- **Tool orchestration**: Selects tools and changes strategies dynamically

Best uses

You can use autonomous agents for the following needs:

- Complex, open-ended tasks with unclear solution paths
- Scenarios requiring real-time adaptation and reasoning
- Environments with high variability or need for personalization

Examples

Consider implementing autonomous agents for the following uses:

- Research agents synthesizing new information
- Adaptive customer support and troubleshooting
- Automation that iteratively refines results based on feedback

Advantages

Autonomous agents offer the following advantages:

- **Highly adaptable**: Handles unforeseen situations
- **Dynamic decision-making**: Iterates and improves over time
- **Reduces human intervention**: Manages complexity autonomously

Limitations

Autonomous agents can also encounter the following challenges:

- **Unpredictable outcomes**: Requires robust monitoring and safeguards
- **Higher complexity and cost**: More difficult to debug and guarantee compliance

---

Now that you are familiar with all three paradigms, review the following summary table to compare these paradigms side-by-side.

Summary table

The following table compares three AI systems, their processes, use cases, and pros and cons.

AI System type	Process	Use Case	Pros	Cons
Single LLM	Input → LLM → Output	Summarization, classification	Simple, fast, low cost	Not adaptable, lacks context
Workflow	Parallel LLMs → Aggregation → Output	Structured multi-step tasks	Predictable, easy to audit	Rigid, not dynamic
Agent	Plan → Act → Observe → (repeat agent loop)	Complex, adaptive automation	Flexible, learns from feedback	Unpredictable, complex, pricier

---

Real-world implementation practices

In practice, hybrid architectures are common. They combine workflow reliability with agent flexibility to achieve the best results.

Recent standards, including Model Context Protocol, or MCP, from Anthropic, and Agent Communication Protocol, or ACP, from IBM, ease integration, monitoring, and governing both approaches at scale.

Key takeaways

When selecting an AI agent, reflect on the following considerations:

- **Start simple**: Use the most straightforward solution that fulfills your needs. For example, you can use single LLM features for atomic needs.
- **Leverage workflows:** When predictability, compliance, and efficiency matter
- **Deploy agents selectively:** Only when adaptability, complex reasoning, or open-ended problem solving are required

---

Author

Faranak Heidari