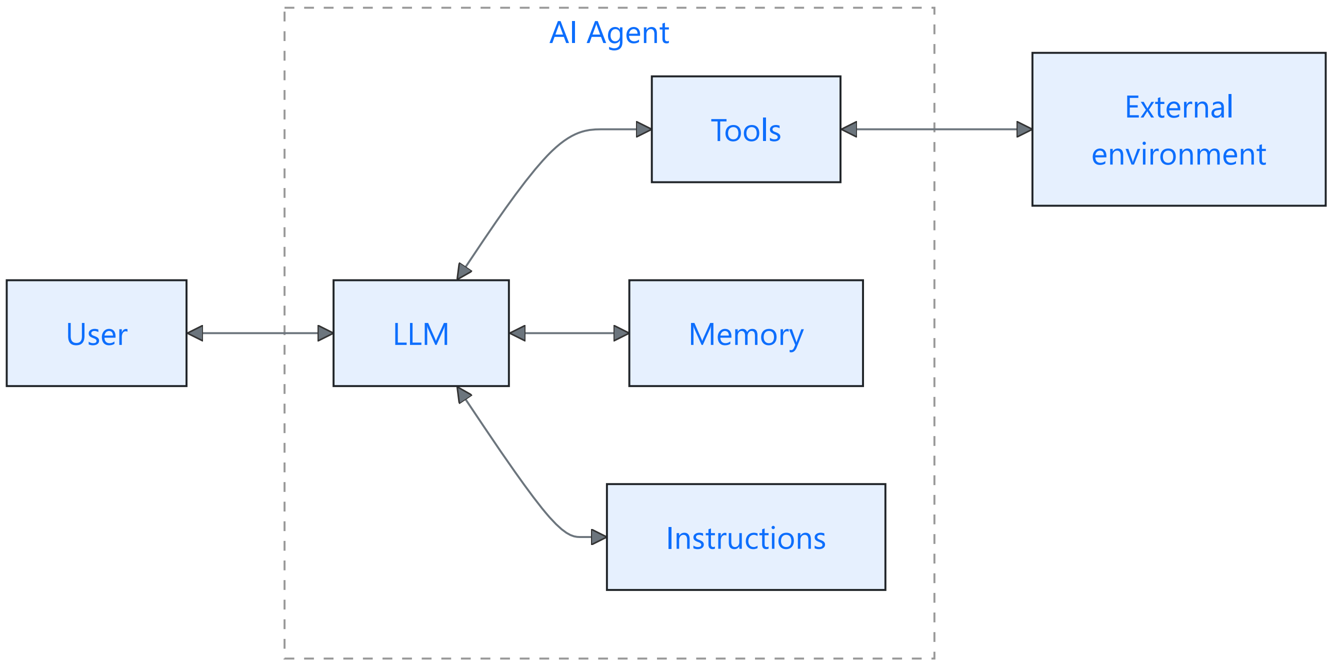
Agentic AI Systems

Introduction: What Are Agentic AI Systems?



An AI agent is a system that can perceive its environment, make decisions, and take actions autonomously to achieve specific goals. An agent can:

1. Understand and decompose complex goals into actionable steps
2. Plan sequences of actions to achieve those goals
3. Use tools and interact with external systems to gather information and make changes
4. Maintain memory and context across multiple interactions
5. Adapt and self-correct when things don't go as planned

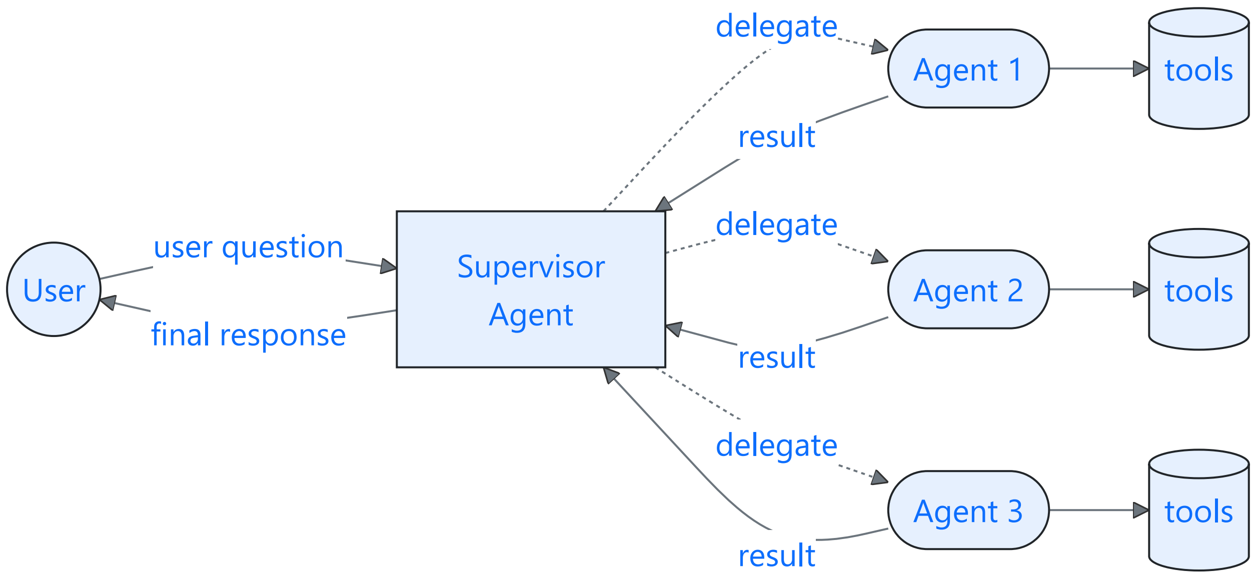
Why are agents important now?

1. Modern LLMs have dramatically improved reasoning capabilities.
2. Function calling and tool use have become standard features.
3. The context window explosion

What's the spectrum from simple to agentic?

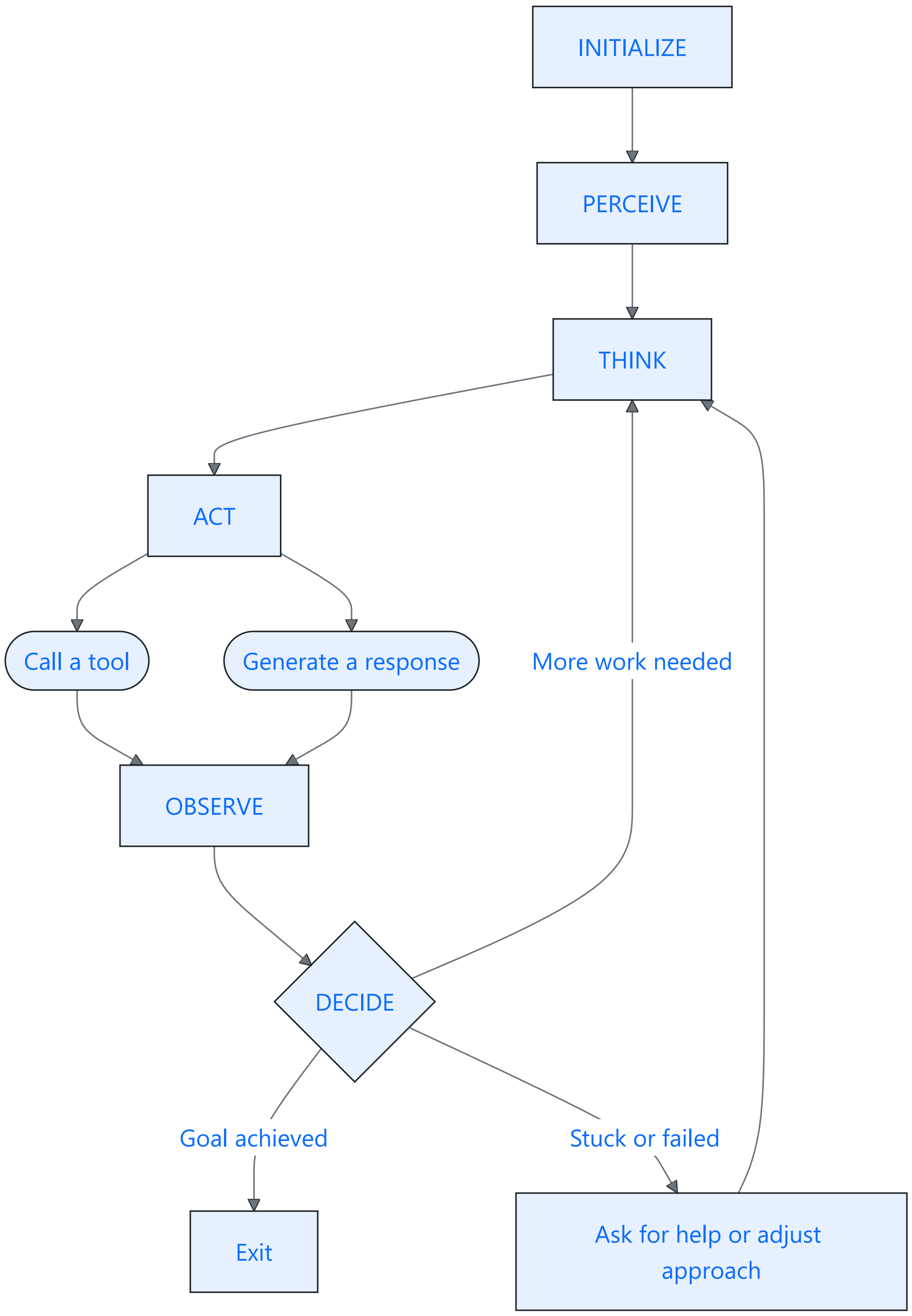
- Level 0 - Simple Chatbot
- Level 1 - Tool-Using Assistant
- Level 2 - Task-Oriented Agent
- Level 3 - Autonomous Agent

Multi-Agent Systems

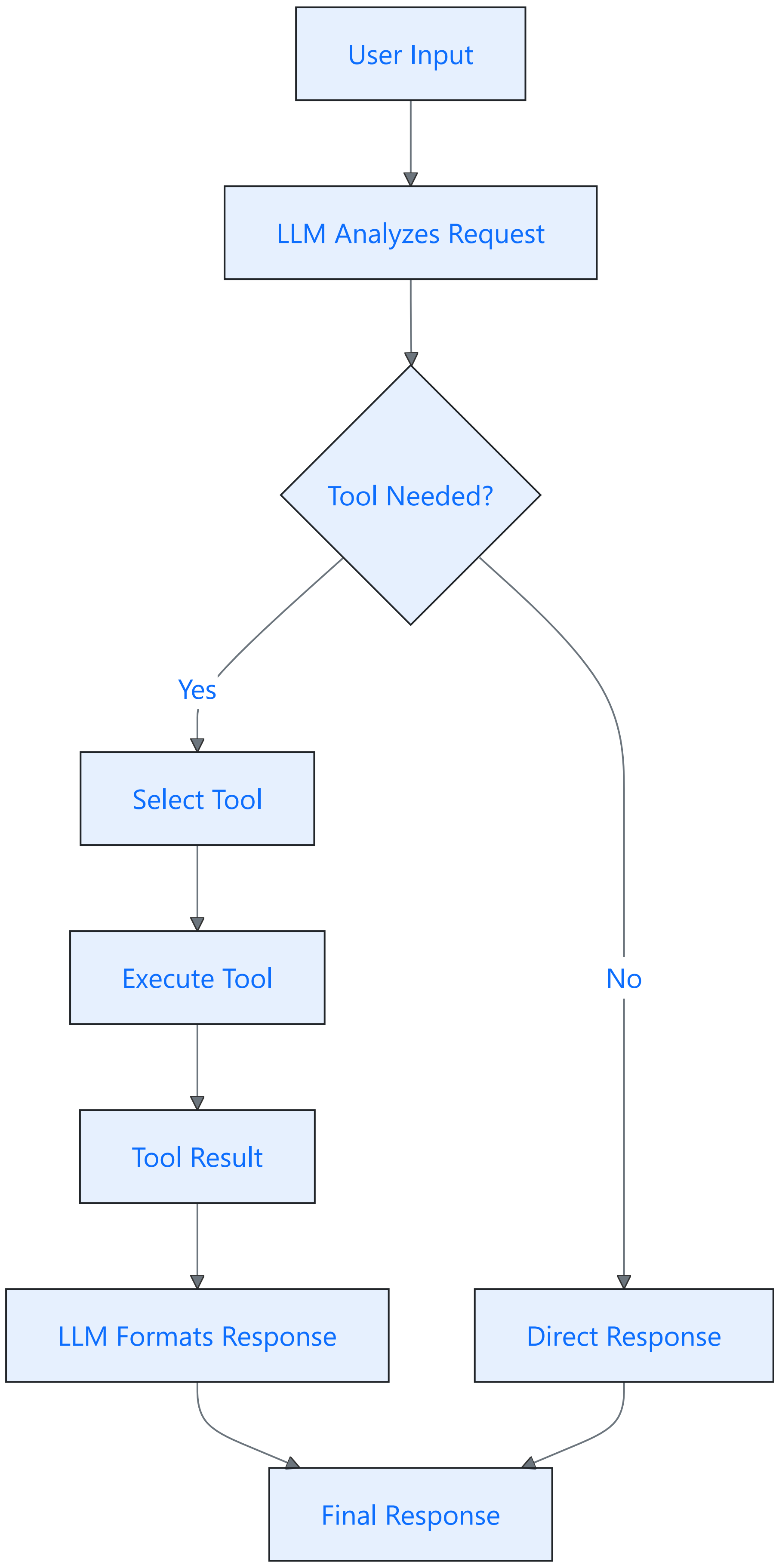


Sometimes one agent isn't enough. Multi-agent systems involve multiple specialized agents working together.

The Basic Agent Loop



Tools and Environment Interaction



How Do Agents Discover and Understand Tools?

This is crucial. The agent needs to know:

1. What the tool does

2. When to use it
3. How to use it
4. What it returns

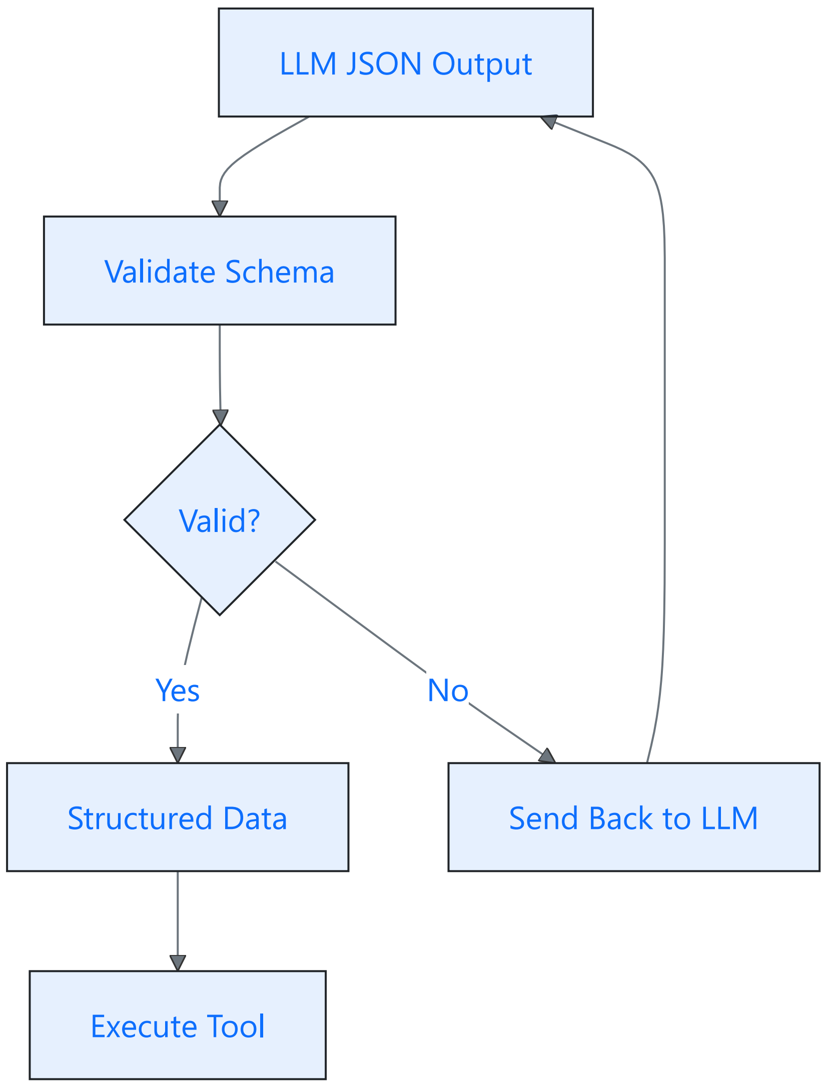
Here's an example tool description:

```json
{
  "name": "search_web",
  "description": "Searches the internet for current information.",
  "parameters": {
    "query": {
      "type": "string",
      "description": "The search query. Be specific and use keywords."
    },
    "num_results": {
      "type": "integer",
      "description": "Number of results to return (1-10)",
      "default": 5
    }
  },
  "returns": "A list of search results with titles, snippets, and URLs"
}
```

What makes a good tool description? It should be:

- Clear and concise
- Action-oriented
- Example-rich
- Constraint-aware

How Do Agents Decide Which Tool to Use?



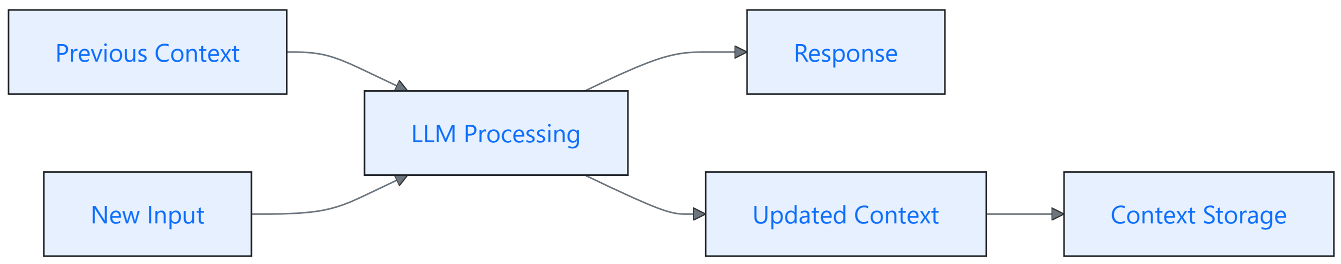
Executing Actions Safely

1. When possible, hardcode the logic.
2. Read-only by default
3. Human in the loop
4. Sandboxing
5. Rate limiting
6. Audit logging

What should be automated vs. human-approved?

- **Automate**: Reading data, searching, analyzing, generating content drafts
- **Require approval**: Sending emails, making purchases, deleting data, publishing content
- **Never automate**: High-stakes decisions (legal, medical, financial), actions affecting many people

Memory and State Management



Types of Memory

1. Short-term (Current Task)
2. Episodic Memory (Past Interactions)
3. Semantic Memory (Learned Knowledge)
4. Procedural Memory (Learned Skills/Patterns)

How Is Agent Memory Implemented?

Strategies:

1. In-Context Memory (Simple)
2. Vector Database (Semantic Search)
3. Structured Storage (Databases, Files)

Best Practice: Use a hybrid approach.

How do you balance cost and capability?

1. Selective memory retrieval
2. Memory summarization
3. Tiered storage
4. Memory pruning

Prompting for Agents

Agent prompts need to establish:

1. The agent's role and persona
2. How to reason and make decisions
3. How to use tools effectively
4. When to stop and ask for help
5. How to handle errors

Example agent system prompt:

```
IDENTITY:

You are a helpful research assistant agent. Your role is to help users
find and synthesize information from various sources.

CAPABILITIES:

You have access to these tools:
```

```
- search_web: Search the internet for current information
- read_document: Read and analyze documents
- execute_code: Run Python code for analysis

DECISION-MAKING PROCESS:

1. Understand the user's goal clearly
2. Break complex tasks into steps
3. Use tools when you need external information or computation
4. Reason about each tool's output before proceeding
5. Synthesize findings into clear, well-organized responses

GUIDELINES:

- Always explain your reasoning before using a tool
- If you're unsure, ask clarifying questions
- If a tool fails, try an alternative approach
- Keep track of what you've learned to avoid repeating work
- Be honest about uncertainty or limitations

ERROR HANDLING:

- If a tool returns an error, explain what went wrong
- Suggest alternatives when possible
- Don't give up after one failure - try different approaches
- If truly stuck, ask the user for guidance
```

Few-shot examples of tool use:

```
Example 1:

User: What's the current stock price of Apple?
Agent Thought: I need current data, so I'll use search_web
Action: search_web("AAPL stock price today")
Observation: [search results showing $178.32]
Response: Apple's stock (AAPL) is currently trading at $178.32.

Example 2:

...
```

These examples teach the agent the expected pattern of behavior.

Evaluation and Debugging

How do you evaluate agent performance?

Metrics that matter:

1. Success Rate
2. Efficiency
3. Cost
4. Quality

5. Robustness

How do you test agents systematically?

Create a **test suite** with diverse scenarios:

```
test_cases = [
    {
        "name": "Simple information retrieval",
        "input": "What's the capital of France?",
        "expected_tools": ["search_web"],
        "success_criteria": "Response contains 'Paris'"
    },
    {
        "name": "Multi-step reasoning",
        "input": "Compare populations of top 3 largest countries",
        "expected_tools": ["search_web", "execute_code"],
        "success_criteria": "Shows Russia, Canada, USA/China with data"
    },
    {
        "name": "Error handling",
        "input": "Read non-existent file data.csv",
        "expected_tools": ["read_document"],
        "success_criteria": "Gracefully handles error, asks for correct filename"
    }
]
```

Run these tests regularly and track success rates over time.

Challenges and Limitations

Current Challenges

1. Reliability and Consistency
2. Cost and Latency
3. Context Window Limitations
4. Long-Horizon Tasks

How Do Agents Fail?

1. Infinite Loops
2. Tool Misuse
3. Hallucinated Actions
4. Premature Giving Up
5. Catastrophic Context Forgetting

Real-World Applications

Current Use Cases

1. Customer Service Automation
2. Software Development Assistants
3. Research and Analysis Tools
4. Personal Productivity Agents
5. Business Process Automation

What Makes a Good Use Case?

1. Repetitive with variations
2. Multi-step but well-defined
3. Information-intensive
4. Low-risk or reversible
5. Time-consuming but low-value