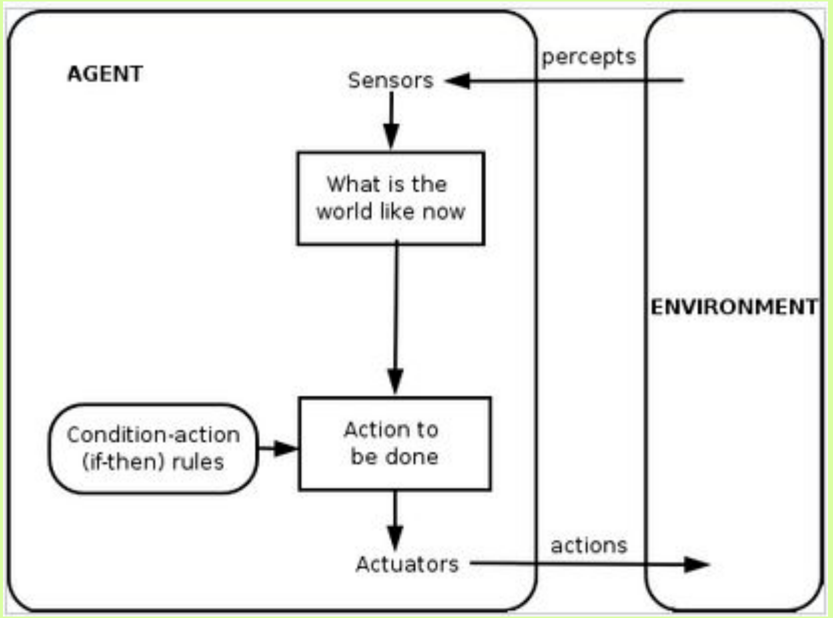
What I learned building my own AI Agent

Windsor Hackforge

Nov 14, 2025

- An entity that perceives its environment, takes action autonomously
- “Agentic AI” proactively pursues goals, making decisions and taking action over extended periods



What’s an AI Agent?

One way of classifying agents is based on their decision-making processes and how they interact with their environment

Types of AI Agent

Simple Reflex

Reacts to current perception with predefined rules.
e.g. thermostat, automatic doors

Model-Based

Maintains internal state about the world and acts on it.
e.g. Roomba

Goal-Based

Acts to achieve specific goals using planning
e.g. Goal-oriented action planning (GOAP) AI in video games

Utility-Based

Maximizes utility function for optimal outcomes
e.g. financial trading bot, recommendation system

Learning

Improves performance through experience
e.g. spam filters, alphaGo

* Russell and Norvig "Artificial Intelligence: A Modern Approach" 4th edition

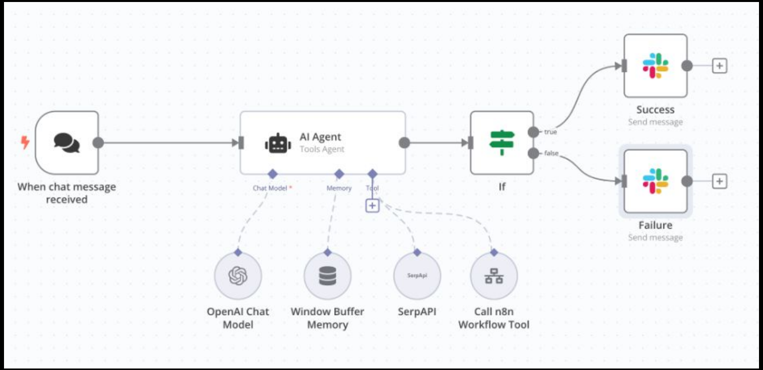
Practical Implementation Approaches

Agent Workflow

- Structured, deterministic processes
- Agents operate within predefined sequences and constraints
- Orchestrated coordination between multiple tools, agents, and environments through explicit patterns
- Tech stack will include orchestration platforms (e.g. **Airflow**), monitoring tools (e.g. **Grafana**) as well as LLM and tools
- This type of agent is used to perform predefined processes

Autonomous Agent

- Self-directed, adaptive processes
- Agents possess greater independence and decision-making authority
- End-to-end execution with minimal human intervention
- Dynamic task decomposition and planning
- Emergent behaviour arises from agent interaction
- Tech stack is more fluid
- This type of agent is used for processes that are more open-ended (e.g. research)



n8n

n8n is a workflow automation platform that purports to “give technical teams the flexibility of code with the speed of no-code.” n8n supports visual development of workflow-style agents.

LangChain

LangChain is the most comprehensive open-source framework for building AI agents. It provides modular building blocks for many agent types and orchestration tools.

Cline

Cline is an interactive, open-source coding agent built for Visual Studio.

Amazon Bedrock

Amazon Bedrock AgentCore provides AI services and infrastructure designed to work independently or in concert with other agent frameworks.

Public Agent Offerings

Building an AI Agent

Implementation Steps

1. Define agent purpose and capabilities

2. Plan, build, test:
   - infrastructure
   - tools
   - orchestration

3. Implement core agent loop

4. Integrate LLM provider

5. Create user interface

Tips:

- **Use conceptual models as design shortcuts**
- **Good infrastructure is the key to going beyond context limit**
- **It's difficult for autonomous agents to hit the bullseye every time; for that, use a workflow**

1 - Send tools & prompt

- **Application Action:** Package available tools as JSON schema, send with user prompt
- **What happens:** OpenAI receives both the conversation and available tools

2 - Model Requests Tools

- **Model Action:** Processes prompt and determines tool choices
- **Response Format:** Returns `finish_reason: “tool_calls”` with function name and argument

3 - Execute Tools & Return

- **Application Action:** Calls actual functions using model-provided parameters
- **What to Return:** Tool execution results as `tool_call_id` + response JSON

Tool Calls with the OpenAI API

4 - Generate Response

- **Model Action:** Synthesizes original prompt, conversation history, and tool results
- **Output:** Natural language response including all gathered information
- **Completion:** `finish_reason: “stop”` when ready

Chaining allows agents to work step-by-step

Each round builds on previous tool results

Application controls conversation flow

*Streaming models can chain tools during the generation process

Tool Chaining

Tool Agent

LLM Integration

Works with local and cloud models via the OpenAI API.

Tool Calling & Tool Chaining

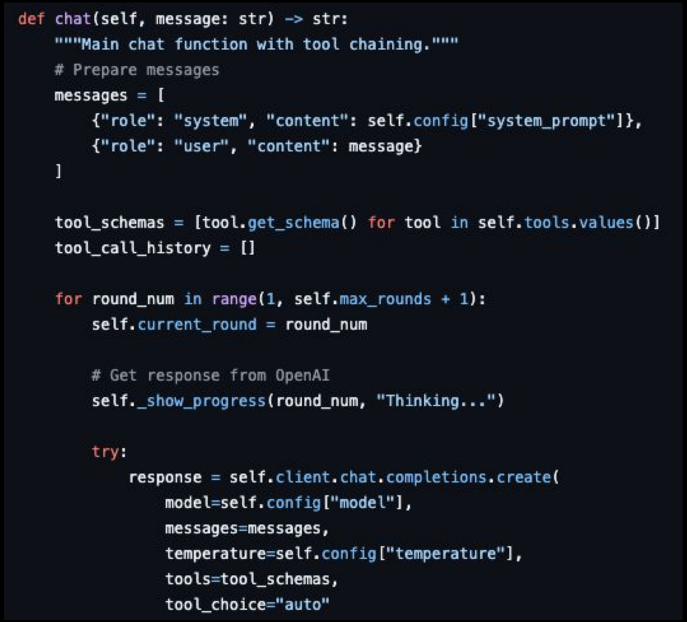
Model autonomously requests tools based on user prompt, and supports multiple rounds of tool calls within a single conversation turn.

CLI Interface

Supports interactive use or accepts prompts via stdin or argument. Tool Agent can also be imported as a Python module.

Easy to extend

Built for simplicity and clarity. Add tools or missing features



```python
def chat(self, message: str) -> str:
    """Main chat function with tool chaining."""
    # Prepare messages
    messages = [
        {"role": "system", "content": self.config["system_prompt"]},
        {"role": "user", "content": message}
    ]

    tool_schemas = [tool.get_schema() for tool in self.tools.values()]
    tool_call_history = []

    for round_num in range(1, self.max_rounds + 1):
        self.current_round = round_num

        # Get response from OpenAI
        self._show_progress(round_num, "Thinking...")

        try:
            response = self.client.chat.completions.create(
                model=self.config["model"],
                messages=messages,
                temperature=self.config["temperature"],
                tools=tool_schemas,
                tool_choice="auto"
```

Source and documentation:

https://github.com/dsartori/tool-agent

Tool Agent Demo

Windsor Hackforge

Nov 14, 2025