INNOVATION AI LAB

INO AI LAB

[AI EDUCATION • COURSE MATERIAL]

Building AI Agents That Actually Work

LangGraph, AutoGen, function calling

LEVEL	DURATION	LESSONS
Advanced	**6 hours**	**10**

[00]

Table of Contents

[LESSON 01]

What an Agent Really Is (and Isn't)

An AI agent is an LLM in a loop with tools and a goal. Each iteration: the model observes state, decides on an action (call a tool, ask the user, or finish), executes, and observes the result. Strip away the buzzwords and that's the entire concept. The interesting engineering is in tool design, state management, error recovery, and knowing when to stop.

Most production 'agents' in 2026 are actually short workflows — 2 to 5 LLM calls with clear branching. True autonomous agents that run for hours unsupervised are still rare and risky outside narrow domains (coding, research, browsing). Start with a workflow, prove value, then expand autonomy only when failure modes are well-understood and recovery is built in.

// KEY TAKEAWAYS

- Agent = LLM + tools + loop + goal.
- Most 'agents' are short workflows; that's fine.
- Expand autonomy only after mapping failure modes.

[LESSON 02]

Tool Design Is the Whole Game

Bad tools make smart models look dumb. Good tool design: clear names ('search_calendar', not 'tool_3'), strict input schemas, helpful error messages that tell the model how to fix the call, and outputs that include only what the model needs. Every extra field in a tool output is context the model has to filter.

Limit the toolset. A model with 5 well-chosen tools outperforms one with 50 mediocre tools — the cognitive load of picking from a long list degrades both speed and accuracy. For complex domains, use hierarchical agents: a router that picks a specialist agent, each of which has its own narrow toolset. This pattern scales to dozens of capabilities without overwhelming any single decision point.

// KEY TAKEAWAYS

- Tool design beats prompt engineering for agents.
- 5 great tools > 50 mediocre ones.
- Hierarchical agents scale capability without confusion.

[LESSON 03]

State, Memory, and Long-Running Tasks

For anything beyond a few turns, you need persistent state. LangGraph, CrewAI, and AutoGen all provide checkpointing — save state to disk or DB after every step so a crash doesn't lose hours of work. Build resumability from day one; retrofitting it is painful. Use a versioned schema for state so you can upgrade agent logic without losing in-progress sessions.

Long-term memory is a separate concern from state. Use a vector store for facts the agent learns about users or systems, with explicit write paths ('remember that the user prefers concise answers'). Avoid letting the agent freely write to memory — it will pollute storage with irrelevant facts. Periodic memory consolidation by a separate process keeps signal high and storage costs bounded.

// KEY TAKEAWAYS

- Checkpoint state every step; build resumability early.
- Versioned schemas survive agent upgrades.
- Gate memory writes; consolidate periodically.

[LESSON 04]

Observability, Evaluation, and Safety

Agents are non-deterministic systems with branching execution paths. Without tracing, debugging is impossible. Use LangSmith, Langfuse, or Arize to log every step — input, tool calls, intermediate reasoning, output. When something goes wrong in production, you need the full trace to fix it. Build this in from day one, not after the first incident.

Evaluation is harder than for one-shot LLM calls. Use end-to-end task success rate on a labeled benchmark, plus per-step metrics (tool-call accuracy, recovery from errors, cost per task). For safety-critical agents, add a separate 'guardian' model that watches actions and can interrupt. Set hard limits on actions: max steps, max spend, max tool calls. Agents without limits are how budgets and reputations die.

// KEY TAKEAWAYS

- Tracing is non-negotiable for agents.
- Evaluate end-to-end AND per-step.
- Hard limits prevent runaway agents.

[NEXT]

Keep Going

You've completed this course material. The real learning starts when you apply what you've read. Pick one idea from this PDF and run a small experiment this week. Document what worked and what didn't. Share your findings with the community.

Explore more free courses, daily AI tips, and curated tools at:

innovationailab.com

Have feedback or want to suggest a topic? We read every message.

hello@innovationailab.com

— Innovation AI Lab Team —

// part of LumiLife Tech