AI Agents

From Theory to Implementation

Dhara Mungra
University of Mannheim; SimPPL

Building AI Agents

- From LLM prompts to goal-directed systems
- Concepts, implementation, and critical thinking
- Outcome: understand what an agent is and build a simple one in Python

Why do we need Agents?

When LLMs Are Enough

- Large Language Models are effective for **well-defined, single-step tasks**
- Examples include:
  - Summarizing a document
  - Rewriting or editing text
  - Answering a question based on provided context
- In these cases:
  - All required information is already available
  - No intermediate reasoning steps are needed
  - No external systems need to be accessed

Why plain LLM calls are often not enough

Many real-world problems are **not single-step tasks**.They require a sequence of actions and decisions.

Example:

> “Find the last three months of sales for our top five products, compare them to the previous quarter, and draft a summary for management.”

To solve this, a system must:

- Retrieve data from a database
- Identify top-performing products
- Perform comparisons across time periods
- Generate a structured summary

A standalone LLM:

- Does not have direct access to data
- Cannot reliably execute multi-step workflows
- Cannot validate intermediate results

What Are AI Agents

AI agents are systems designed to handle **multi-step, goal-oriented tasks**.

They:

- Break problems into smaller steps
- Decide what actions to take
- Use tools to access data or perform operations
- Iterate until the task is complete

According to IBM: “AI agents are systems that can act on behalf of a user using workflows and tools”

LLM vs Agent

Plain LLM usage	Agentic system
You give one prompt	The system manages multiple steps
Mostly returns text	Can retrieve data, call tools, and decide next actions
Usually no real memory beyond supplied context	Can maintain state across turns or steps
User does all orchestration	System handles part of the orchestration

Example

- Prompt only: “Write an SQL query to show last month’s failed payments.”
- Agent: Understand the task, inspect schema, generate query, run query safely, handle errors, summarize results.

Agent Workflow

Content:

To understand agents, it helps to break them into parts:

1. Goal — what the user wants done
2. Instructions — how the system should behave
3. Model — the reasoning/generation engine
4. Memory/state — what has happened so far
5. Tools/skills — what the system can do beyond text generation
6. Workflow/controller — how the next action is chosen
7. Evaluation/guardrails — how output is checked

Agent Workflow Example

User asks: “Summarize our support tickets from this week and identify the top product issue.”

The agent:

- retrieves tickets,
- groups similar complaints,
- summarizes the main issue,
- then formats a response for the manager.

Agent Workflow Example

User: “Why is the office Wi-Fi failing?”

Possible workflow:

- Ask clarifying questions,
- Check network logs,
- Inspect whether recent changes occurred,
- Summarize likely cause,
- Propose next actions.

Where prompts fit in all this?

Prompts matter, but prompting alone is not agent design

- Prompts control **how the model behaves in a single step**
- A well-designed prompt can define:
  - Role
  - Task
  - Constraints
  - Output format
- However, prompts alone do **not provide**:
  - Memory across interactions
  - Access to tools or external data
  - Control over multi-step workflows

Prompts matter, but prompting alone is not agent design

Weak prompt

“Explain customer churn.”

Stronger prompt inside a workflow

“You are a retention analyst. Using the churn table summary below, explain the top three drivers of churn in bullet points, mention uncertainty if the evidence is weak, and keep the answer under 120 words.”

What are tools and skills and why do they matter?

Why Agents Need Tools

Large Language Models are powerful at generating text, but many real-world tasks require capabilities beyond what the model can reliably provide.

These include:

- Access to **current or real-time information**
- Interaction with **private or proprietary data**
- **Accurate computation** and numerical operations
- **Execution of code or structured queries**
- Integration with **APIs and databases**

Without tools, the model can only **simulate answers**.

With tools, the system can **perform actions and retrieve reliable results**.

Why Tools Are Necessary

In practice, relying only on text generation is often insufficient:

- A model can generate an SQL query,
  but a **database system is required to execute it safely and correctly**
- A model can describe a calculation,
  but a **calculator tool ensures precision and avoids errors**
- A model can approximate exchange rates,
  but a **live API is needed for accurate, up-to-date values**

Key Idea:

Tools provide **execution and reliability**, not just suggestions.

Skills vs Tools

Tools and skills are closely related, but operate at different levels of abstraction.

- A **tool** is a specific, callable capability
  (e.g., a function, API, or database query)
- A **skill** is a higher-level behavior that combines:
  - One or more tools
  - Prompting strategies
  - Workflow logic

In simple terms:

- Tools are **building blocks**
- Skills are **useful behaviors built from those blocks**

Examples of Tools and Skills

Examples of Tools:

- SQL query execution function
- Web search API
- Calculator or Python function

Examples of Skills:

- *Competitive research assistant*:
  searches, filters, summarizes, and structures insights
- *Meeting preparation assistant*:
  processes notes, identifies priorities, and drafts agendas

Agents Can Fail In Predictable Ways

Even a well-designed agent can fail. Common problems include:

- They can hallucinate facts or sources
- They can use the wrong tool or wrong data
- They can misunderstand the task
- They can sound confident even when evidence is weak
- They can lose coherence over many steps

Example:
An agent asked to summarize company data might invent missing numbers instead of admitting the dataset is incomplete.

How To Make Agents More Reliable

Common mitigation strategies include:

- Using retrieval instead of relying on memory alone,
- Narrowing the task,
- Validating tool outputs,
- Asking the model to express uncertainty,
- Logging intermediate steps,
- Keeping humans in the loop for high-stakes decisions,
- Evaluating with test cases rather than intuition.

Example
Instead of asking “What are our top support issues?”, retrieve last week’s ticket data first and instruct the model to answer only from that evidence.

Different Types Of Agents And Where They Are Used

- Conversational agents — customer support, education, internal assistants
- Coding agents — code generation, debugging, refactoring, test writing
- Task agents — scheduling, reporting, workflow orchestration
- Personal information management agents — email, notes, calendar, reminders
- Domain agents — finance, legal, sales, operations, research

Why This Matters For You

- You will use agentic products in industry
- But client work requires control over data, workflow, and outputs
- Building your own agent lets you adapt to domain-specific needs
- Understanding the architecture helps you judge what commercial tools can and cannot do

Lets build an AI Agent

- A simple conversational agent
- A clear system prompt
- Message history as memory
- One tool/skill extension point
- A small set of failure tests

Feedback!

Ask Me Anything!

Questions?
Please reach out to
dhara.atul.mungra@uni-mannheim.de

Thank you!