1 Introduction to agents and their world

This chapter covers

- Defining the concept of agents
- Differentiating the components of an agent
- Analyzing the rise of the agent era: Why agents?
- Peeling back the AI interface
- Navigating the agent landscape

The agent isn't a new concept in machine learning and artificial intelligence (AI). In reinforcement learning, for instance, the word *agent* denotes an active decision-making and learning intelligence. In other areas, the word *agent* aligns more with an automated application or software that does something on your behalf.

1.1 Defining agents

You can consult any online dictionary to find the definition of an agent. The *Merriam-Webster Dictionary* defines it this way (www.merriam-webster.com/dictionary/agent):

- One that acts or exerts power
- Something that produces or can produce an effect
- A means or instrument by which a guiding intelligence achieves a result

The word *agent* in our journey to build powerful agents in this book uses this dictionary definition. That also means the term *assistant* will be synonymous with *agent.* Tools like OpenAI's GPT Assistants will also fall under the AI agent blanket. OpenAI avoids the word *agent* because of the history of machine learning, where an agent is self-deciding and autonomous.

Figure 1.1 shows four cases where a user may interact with a large language model (LLM) directly or through an agent/assistant proxy, an agent/assistant, or an autonomous agent. These four use cases are highlighted in more detail in this list:

- *Direct user interaction*—If you used earlier versions of ChatGPT, you experienced direct interaction with the LLM. There is no proxy agent or other assistant interjecting on your behalf.
- *Agent/assistant proxy*—If you've used Dall-E 3 through ChatGPT, then you've experienced a proxy agent interaction. In this use case, an LLM interjects your requests and reformulates them in a format better designed for the task. For example, for image generation, ChatGPT better formulates the prompt. A proxy agent is an everyday use case to assist users with unfamiliar tasks or models.
- *Agent/assistant*—If you've ever used a ChatGPT plugin or GPT assistant, then you've experienced this use case. In this case, the LLM is aware of the plugin or assistant functions and prepares to make calls to this plugin/function. However, before making a call, the LLM requires user approval. If approved, the plugin or function is executed, and the results are returned to the LLM. The LLM then wraps this response in natural language and returns it to the user.
- *Autonomous agent*—In this use case, the agent interprets the user's request, constructs a plan, and identifies decision points. From this, it executes the steps in the plan and makes the required decisions independently. The agent may request user feedback after certain milestone tasks, but it's often given free rein to explore and learn if possible. This agent poses the most ethical and safety concerns, which we'll explore later.

Figure 1.1 demonstrates the use cases for a single flow of actions on an LLM using a single agent. For more complex problems, we often break agents into profiles or personas. Each agent profile is given a specific task and executes that task with specialized tools and knowledge.

Multi-agent systems are agent profiles that work together in various configurations to solve a problem. Figure 1.2 demonstrates an example of a multi-agent system using three agents: a controller or proxy and two profile agents as workers controlled by the proxy. The coder profile on the left writes the code the user requests; on the right is a

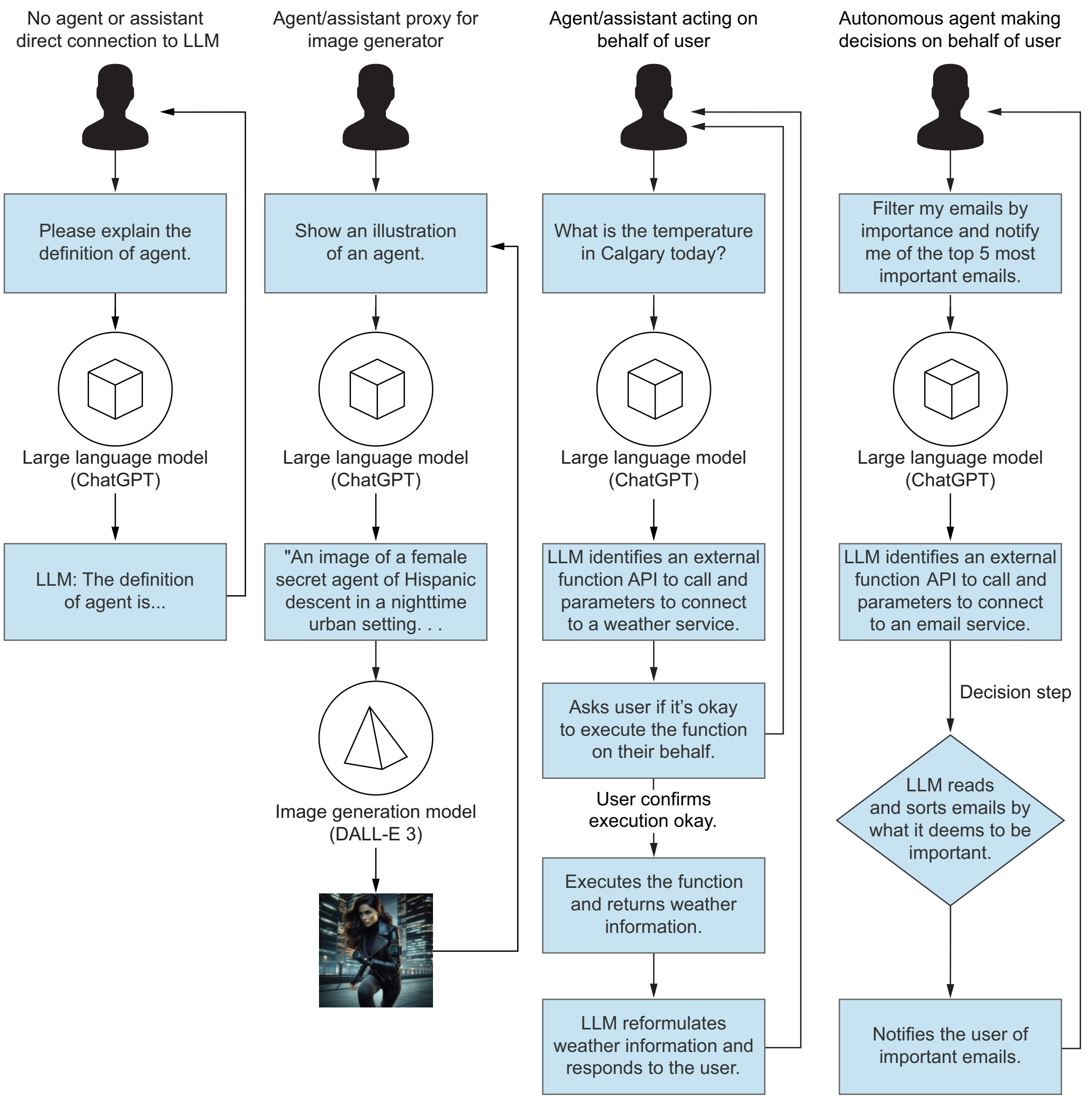

No agent or assistant direct connection to LLM
Agent/assistant proxy for image generator
Agent/assistant acting on behalf of user
Autonomous agent making decisions on behalf of user
Please explain the definition of agent.
Show an illustration of an agent.
What is the temperature in Calgary today?
Filter my emails by importance and notify me of the top 5 most important emails.
Large language model (ChatGPT)
Large language model (ChatGPT)
Large language model (ChatGPT)
Large language model (ChatGPT)
LLM: The definition of agent is...
"An image of a female secret agent of Hispanic descent in a nighttime urban setting. . .
LLM identifies an external function API to call and parameters to connect to a weather service.
LLM identifies an external function API to call and parameters to connect to an email service.
Image generation model (DALL-E 3)
Asks user if it's okay to execute the function on their behalf.
User confirms execution okay.
Executes the function and returns weather information.
LLM reformulates weather information and responds to the user.
Decision step
LLM reads and sorts emails by what it deems to be important.
Notifies the user of important emails.

Figure 1.1 The differences between the LLM interactions from direct action compared to using proxy agents, agents, and autonomous agents

tester profile designed to write unit tests. These agents work and communicate together until they are happy with the code and then pass it on to the user.

Figure 1.2 shows one of the possibly infinite agent configurations. (In chapter 4, we'll explore Microsoft's open source platform, AutoGen, which supports multiple configurations for employing multi-agent systems.)

Multi-agent systems can work autonomously but may also function guided entirely by human feedback. The benefits of using multiple agents are like those of a single

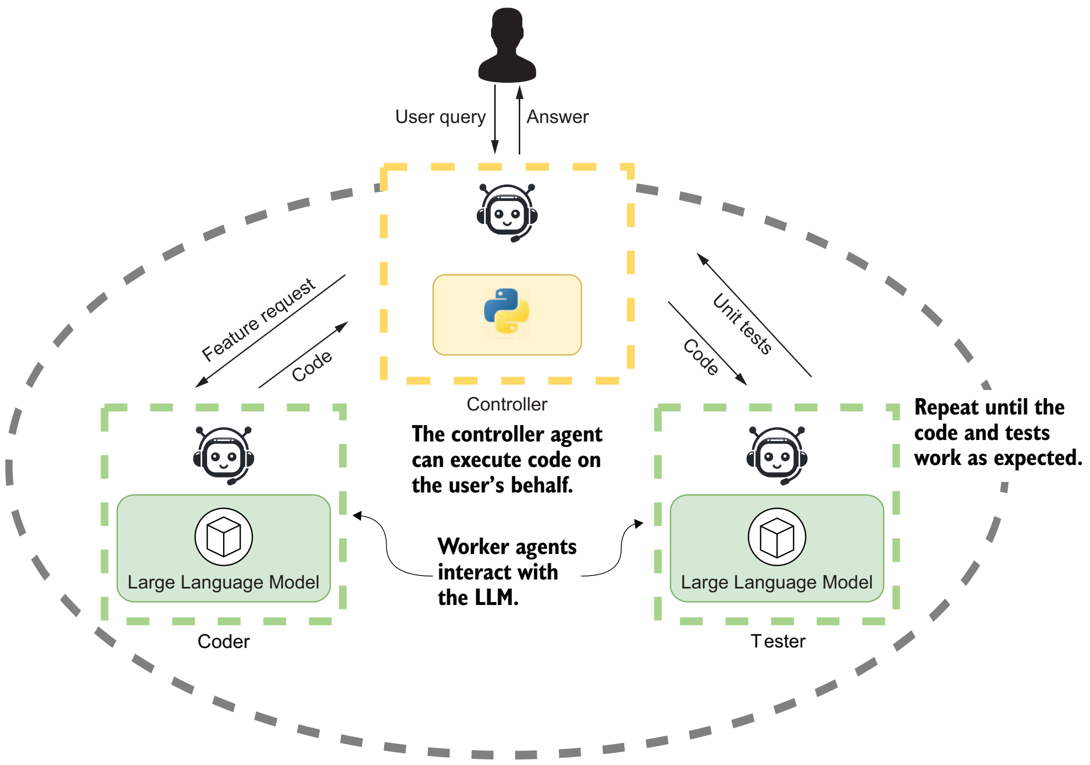

User query
Answer
Feature request
Code
Unit tests
Code
Controller
The controller agent can execute code on the user's behalf.
Repeat until the code and tests work as expected.
Large Language Model
Large Language Model
Worker agents interact with the LLM.
Coder
Tester

Figure 1.2 In this example of a multi-agent system, the controller or agent proxy communicates directly with the user. Two agents—a coder and a tester—work in the background to create code and write unit tests to test the code.

agent but often magnified. Where a single agent typically specializes in a single task, multi-agent systems can tackle multiple tasks in parallel. Multiple agents can also provide feedback and evaluation, reducing errors when completing assignments.

As we can see, an AI agent or agent system can be assembled in multiple ways. However, an agent itself can also be assembled using multiple components. In the next section, we'll cover topics ranging from an agent's profile to the actions it may perform, as well as memory and planning.

1.2 Understanding the component systems of an agent

Agents can be complex units composed of multiple component systems. These components are the tools the agent employs to help it complete its goal or assigned tasks and even create new ones. Components may be simple or complex systems, typically split into five categories.

Figure 1.3 describes the major categories of components a single-agent system may incorporate. Each element will have subtypes that can define the component's type, structure, and use. At the core of all agents is the profile and persona; extending from that are the systems and functions that enhance the agent.

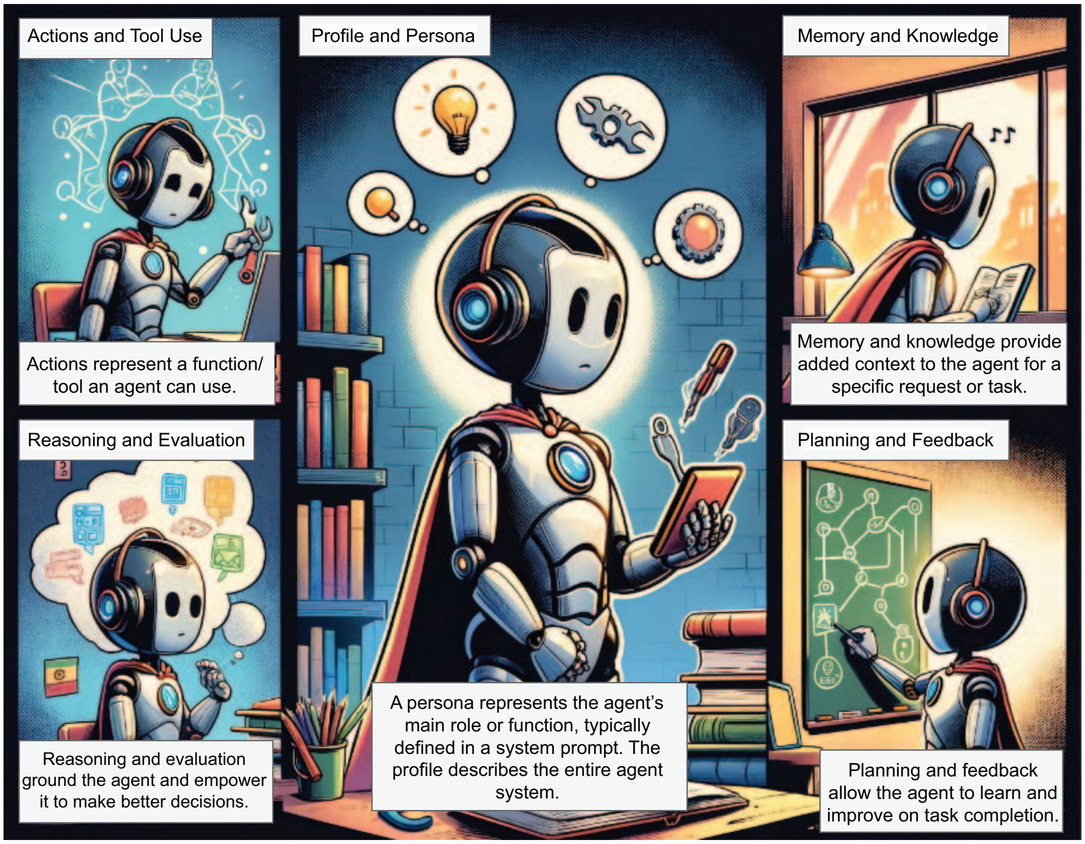

Actions and Tool Use
Actions represent a function/ tool an agent can use.
Reasoning and Evaluation
Reasoning and evaluation ground the agent and empower it to make better decisions.
Profile and Persona
A persona represents the agent's main role or function, typically defined in a system prompt. The profile describes the entire agent system.
Memory and Knowledge
Memory and knowledge provide added context to the agent for a specific request or task.
Planning and Feedback
Planning and feedback allow the agent to learn and improve on task completion.

Figure 1.3 The five main components of a single-agent system (image generated through DALL-E 3)

The agent profile and persona shown in figure 1.4 represent the base description of the agent. The persona—often called the *system prompt*—guides an agent to complete tasks, learn how to respond, and other nuances. It includes elements such as the background (e.g., coder, writer) and demographics, and it can be generated through methods such as handcrafting, LLM assistance, or data-driven techniques, including evolutionary algorithms.

We'll explore how to create effective and specific agent profiles/personas through techniques such as rubrics and grounding. In addition, we'll explain the aspects of human-formulated versus AI-formulated (LLM) profiles, including innovative techniques using data and evolutionary algorithms to build profiles.

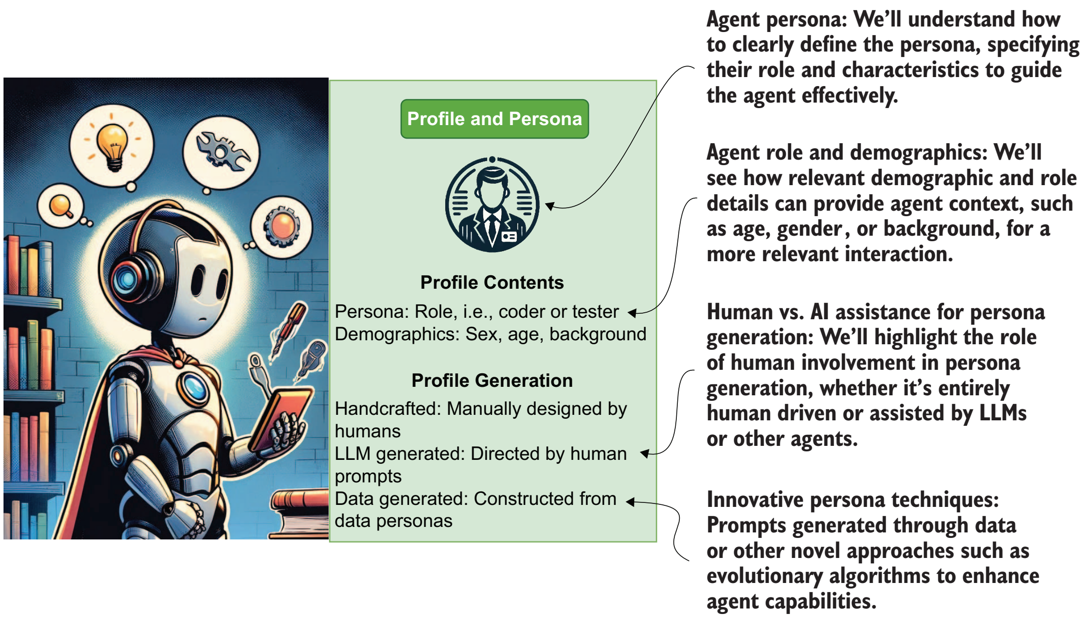

Profile and Persona
Profile Contents
Persona: Role, i.e., coder or tester
Demographics: Sex, age, background
Profile Generation
Handcrafted: Manually designed by humans
LLM generated: Directed by human prompts
Data generated: Constructed from data personas
Agent persona: We'll understand how to clearly define the persona, specifying their role and characteristics to guide the agent effectively.
Agent role and demographics: We'll see how relevant demographic and role details can provide agent context, such as age, gender, or background, for a more relevant interaction.
Human vs. AI assistance for persona generation: We'll highlight the role of human involvement in persona generation, whether it's entirely human driven or assisted by LLMs or other agents.
Innovative persona techniques: Prompts generated through data or other novel approaches such as evolutionary algorithms to enhance agent capabilities.

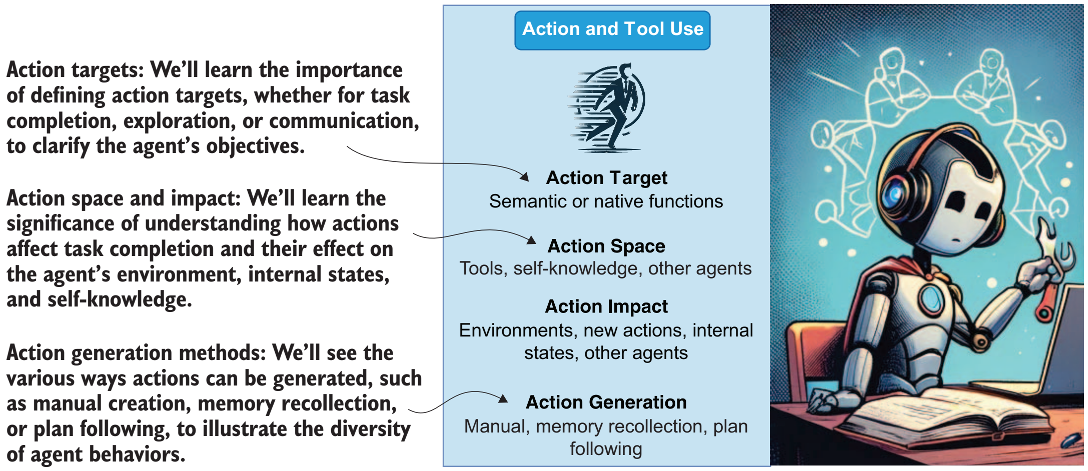
Figure 1.4 An in-depth look at how we'll explore creating agent profiles

NOTE The agent or assistant profile is composed of elements, including the persona. It may be helpful to think of profiles describing the work the agent/assistant will perform and the tools it needs.

Figure 1.5 demonstrates the component actions and tool use in the context of agents involving activities directed toward task completion or acquiring information. These actions can be categorized into task completion, exploration, and communication, with varying levels of effect on the agent's environment and internal states. Actions can be generated manually, through memory recollection, or by following predefined plans, influencing the agent's behavior and enhancing learning.

Understanding the action target helps us define clear objectives for task completion, exploration, or communication. Recognizing the action effect reveals how actions influence task outcomes, the agent's environment, and its internal states, contributing to efficient decision making. Lastly, grasping action generation methods equips us with the knowledge to create actions manually, recall them from memory, or follow predefined plans, enhancing our ability to effectively shape agent behavior and learning processes.

Figure 1.6 shows the component knowledge and memory in more detail. Agents use knowledge and memory to annotate context with the most pertinent information while limiting the number of tokens used. Knowledge and memory structures can be unified, where both subsets follow a single structure or hybrid structure involving a mix of different retrieval forms. Knowledge and memory formats can vary widely from


Action and Tool Use
Action Target
Semantic or native functions
Action Space
Tools, self-knowledge, other agents
Action Impact
Environments, new actions, internal states, other agents
Action Generation
Manual, memory recollection, plan following

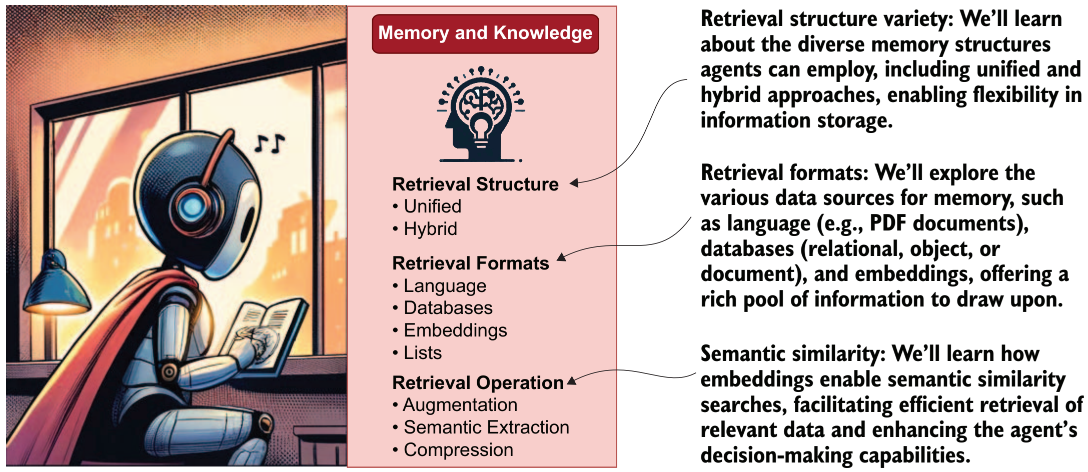
Action targets: We'll learn the importance of defining action targets, whether for task completion, exploration, or communication, to clarify the agent's objectives.

Action space and impact: We'll learn the significance of understanding how actions affect task completion and their effect on the agent's environment, internal states, and self-knowledge.

Action generation methods: We'll see the various ways actions can be generated, such as manual creation, memory recollection, or plan following, to illustrate the diversity of agent behaviors.

Figure 1.5 The aspects of agent actions we'll explore in this book

language (e.g., PDF documents) to databases (relational, object, or document) and embeddings, simplifying semantic similarity search through vector representations or even simple lists serving as agent memories.



Retrieval structure variety: We'll learn about the diverse memory structures agents can employ, including unified and hybrid approaches, enabling flexibility in information storage.

Retrieval formats: We'll explore the various data sources for memory, such as language (e.g., PDF documents), databases (relational, object, or document), and embeddings, offering a rich pool of information to draw upon.

Semantic similarity: We'll learn how embeddings enable semantic similarity searches, facilitating efficient retrieval of relevant data and enhancing the agent's decision-making capabilities.

Figure 1.6 Exploring the role and use of agent memory and knowledge

Figure 1.7 shows the reasoning and evaluation component of an agent system. Research and practical applications have shown that LLMs/agents can effectively reason. Reasoning and evaluation systems annotate an agent's workflow by providing an ability to think through problems and evaluate solutions.

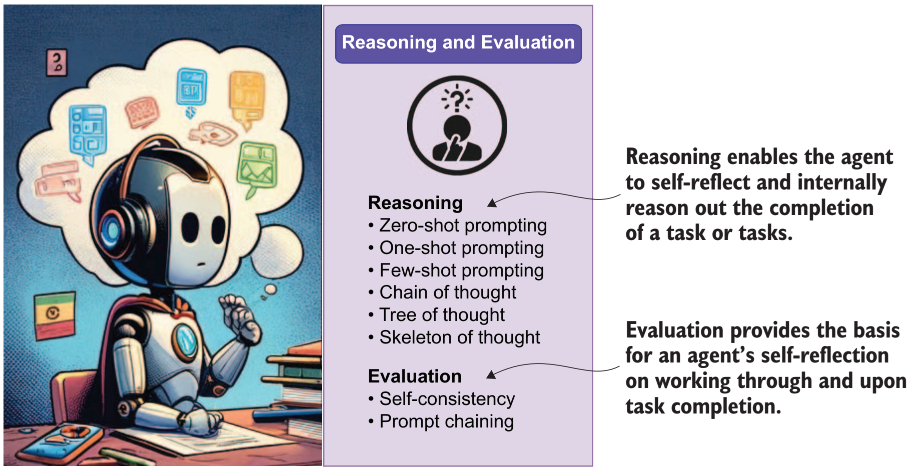

Reasoning and Evaluation
Reasoning
• Zero-shot prompting
• One-shot prompting
• Few-shot prompting
• Chain of thought
• Tree of thought
• Skeleton of thought
Evaluation
• Self-consistency
• Prompt chaining
Reasoning enables the agent to self-reflect and internally reason out the completion of a task or tasks.
Evaluation provides the basis for an agent's self-reflection on working through and upon task completion.

Figure 1.7 The reasoning and evaluation component and details

Figure 1.8 shows the component agent planning/feedback and its role in organizing tasks to achieve higher-level goals. It can be categorized into these two approaches:

- *Planning without feedback*—Autonomous agents make decisions independently.
- *Planning with feedback*—Monitoring and modifying plans is based on various sources of input, including environmental changes and direct human feedback.

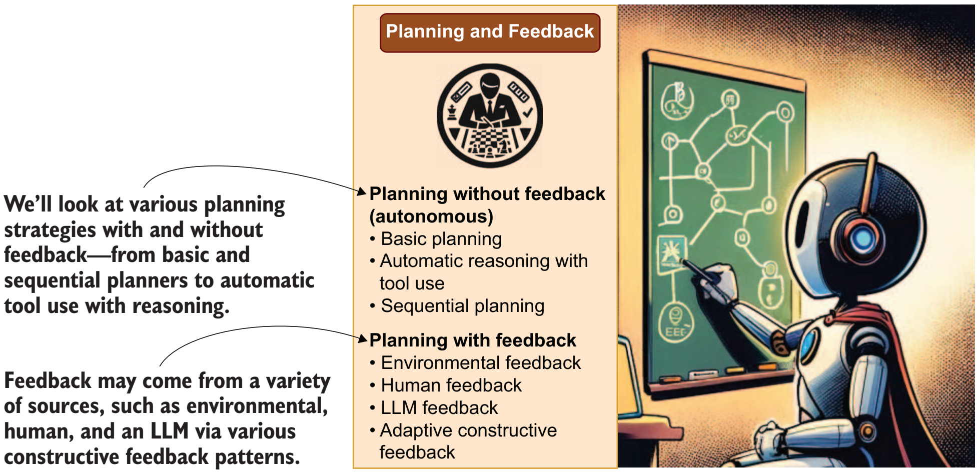

Planning and Feedback
Planning without feedback (autonomous)
• Basic planning
• Automatic reasoning with tool use
• Sequential planning
Planning with feedback
• Environmental feedback
• Human feedback
• LLM feedback
• Adaptive constructive feedback
We'll look at various planning strategies with and without feedback—from basic and sequential planners to automatic tool use with reasoning.
Feedback may come from a variety of sources, such as environmental, human, and an LLM via various constructive feedback patterns.

Figure 1.8 Exploring the role of agent planning and reasoning

Within planning, agents may employ *single-path* reasoning, *sequential reasoning* through each step of a task, or *multipath* reasoning to explore multiple strategies and save the efficient ones for future use. External planners, which can be code or other agent systems, may also play a role in orchestrating plans.

Any of our previous agent types—the proxy agent/assistant, agent/assistant, or autonomous agent—may use some or all of these components. Even the planning component has a role outside of the autonomous agent and can effectively empower even the regular agent.

1.3 Examining the rise of the agent era: Why agents?

AI agents and assistants have quickly moved from the main commodity in AI research to mainstream software development. An ever-growing list of tools and platforms assist in the construction and empowerment of agents. To an outsider, it may all seem like hype intended to inflate the value of some cool but overrated technology.

During the first few months after ChatGPT's initial release, a new discipline called *prompt engineering* was formed: users found that using various techniques and patterns in their prompts allowed them to generate better and more consistent output. However, users also realized that prompt engineering could only go so far.

Prompt engineering is still an excellent way to interact directly with LLMs such as ChatGPT. Over time, many users discovered that effective prompting required iteration, reflection, and more iteration. The first agent systems, such as AutoGPT, emerged from these discoveries, capturing the community's attention.

Figure 1.9 shows the original design of AutoGPT, one of the first autonomous agent systems. The agent is designed to iterate a planned sequence of tasks that it defines by looking at the user's goal. Through each task iteration of steps, the agent evaluates the goal and determines if the task is complete. If the task isn't complete, the agent may replan the steps and update the plan based on new knowledge or human feedback.

AutoGPT became the first example to demonstrate the power of using task planning and iteration with LLM models. From this and in tandem, other agent systems and frameworks exploded into the community using similar planning and task iteration systems. It's generally accepted that planning, iteration, and repetition are the best processes for solving complex and multifaceted goals for an LLM.

However, autonomous agent systems require trust in the agent decision-making process, the guardrails/evaluation system, and the goal definition. Trust is also something that is acquired over time. Our lack of trust stems from our lack of understanding of an autonomous agent's capabilities.

NOTE Artificial general intelligence (AGI) is a form of intelligence that can learn to accomplish any task a human can. Many practitioners in this new world of AI believe an AGI using autonomous agent systems is an attainable goal.

For this reason, many of the mainstream and production-ready agent tools aren't autonomous. However, they still provide a significant benefit in managing and automating

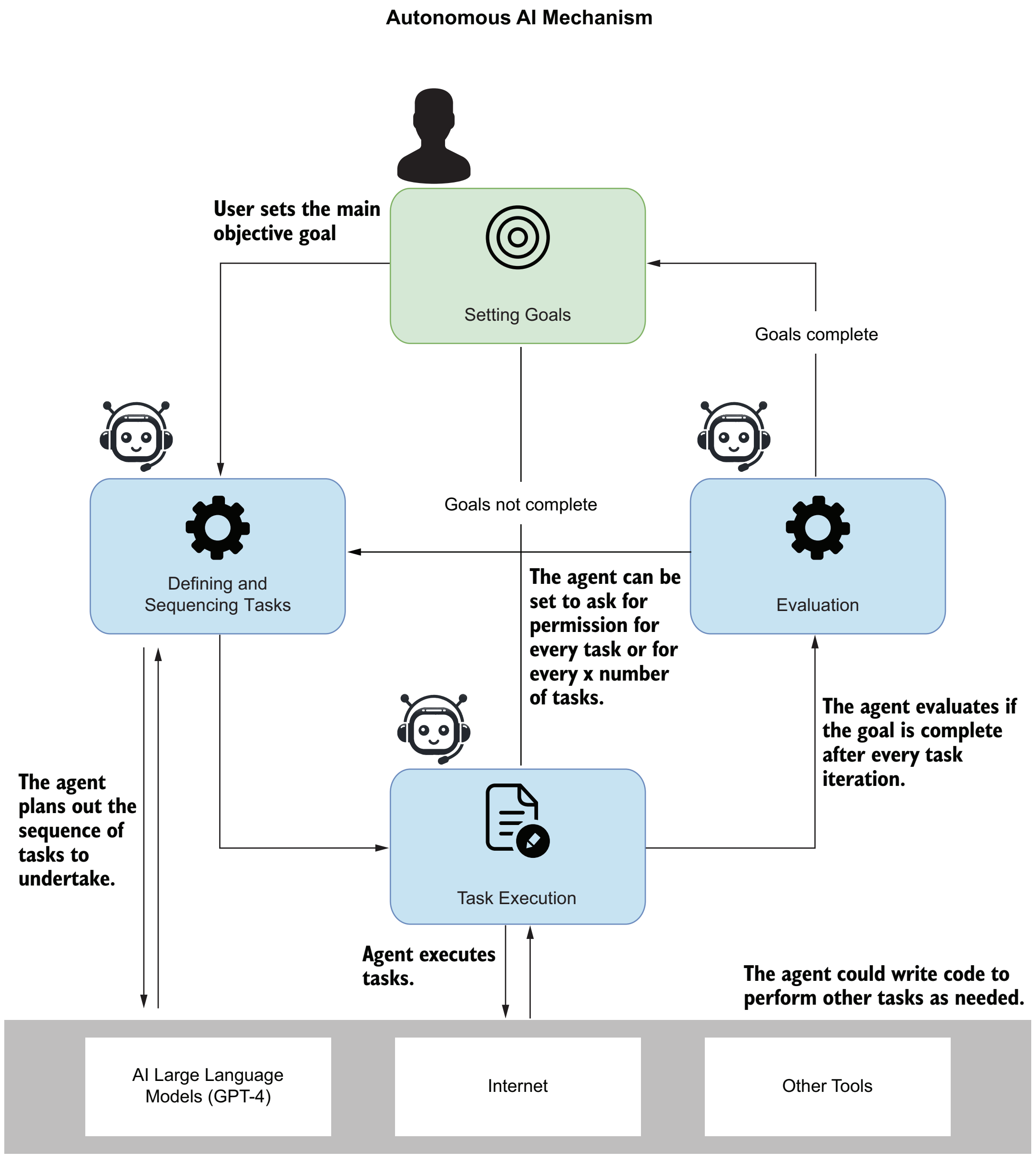

Autonomous AI Mechanism
User sets the main objective goal
Setting Goals
Goals complete
Goals not complete
Defining and Sequencing Tasks
Evaluation
The agent can be set to ask for permission for every task or for every x number of tasks.
The agent evaluates if the goal is complete after every task iteration.
The agent plans out the sequence of tasks to undertake.
Task Execution
Agent executes tasks.
The agent could write code to perform other tasks as needed.
AI Large Language Models (GPT-4)
Internet
Other Tools

Figure 1.9 The original design of the AutoGPT agent system

tasks using GPTs (LLMs). Therefore, as our goal in this book is to understand all agent forms, many more practical applications will be driven by non-autonomous agents.

Agents and agent tools are only the top layer of a new software application development paradigm. We'll look at this new paradigm in the next section.

1.4 Peeling back the AI interface

The AI agent paradigm is not only a shift in how we work with LLMs but is also perceived as a shift in how we develop software and handle data. Software and data will no longer be interfaced using user interfaces (UIs), application programming interfaces (APIs), and specialized query languages such as SQL. Instead, they will be designed to be interfaced using natural language.

Figure 1.10 shows a high-level snapshot of what this new architecture may look like and what role AI agents play. Data, software, and applications adapt to support semantic, natural language interfaces. These AI interfaces allow agents to collect data and interact with software applications, even other agents or agent applications. This represents a new shift in how we interact with software and applications.

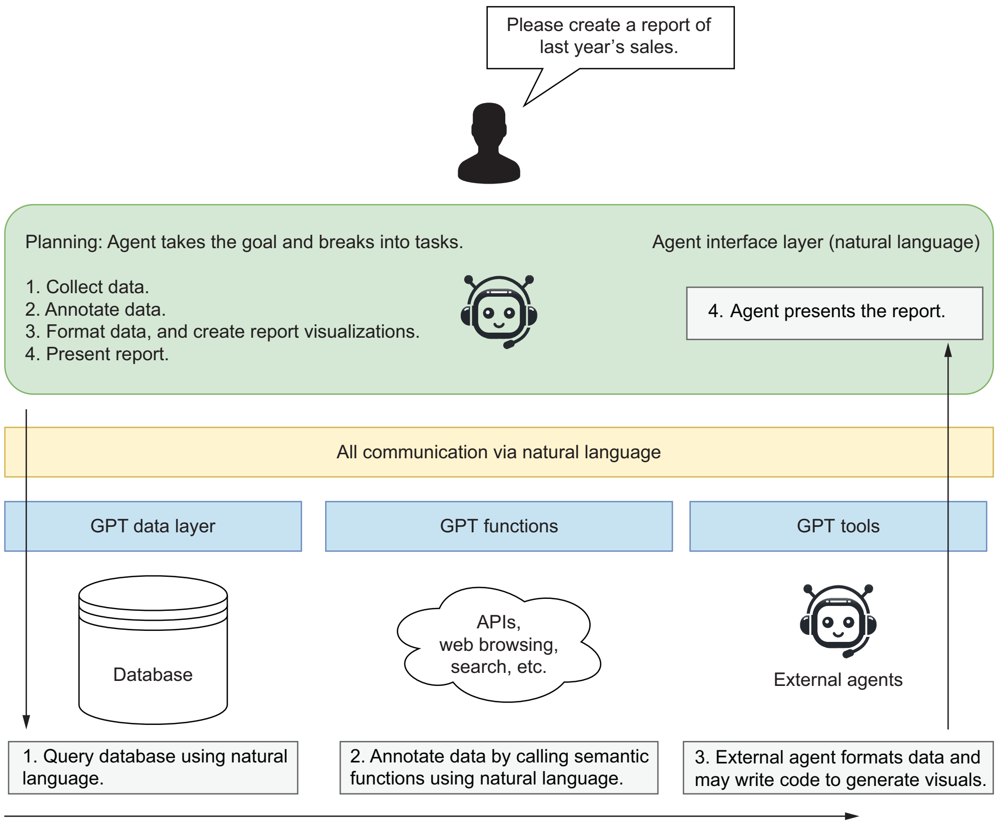

Please create a report of last year's sales.
Planning: Agent takes the goal and breaks into tasks.
1. Collect data.
2. Annotate data.
3. Format data, and create report visualizations.
4. Present report.
Agent interface layer (natural language)
4. Agent presents the report.
All communication via natural language
GPT data layer
GPT functions
GPT tools
Database
APIs, web browsing, search, etc.
External agents
1. Query database using natural language.
2. Annotate data by calling semantic functions using natural language.
3. External agent formats data and may write code to generate visuals.

Figure 1.10 A vision of how agents will interact with software systems

An *AI interface* is a collection of functions, tools, and data layers that expose data and applications by natural language. In the past, the word *semantic* has been heavily used to describe these interfaces, and even some tools use the name; however, "semantic" can also have a variety of meanings and uses. Therefore, in this book, we'll use the term *AI interface*.

The construction of AI interfaces will empower agents that need to consume the services, tools, and data. With this empowerment will come increased accuracy in completing tasks and more trustworthy and autonomous applications. While an AI interface may not be appropriate for all software and data, it will dominate many use cases.

1.5 Navigating the agent landscape

GPT agents represent an entire shift in how consumers and developers approach everything, from finding information to building software and accessing data. Almost daily, a new agent framework, component, or interface pops up on GitHub or in a research paper. This can be overwhelming and intimidating to the new user trying to grasp what agent systems are and how to use them.

Summary

- An agent is an entity that acts or exerts power, produces an effect, or serves as a means for achieving a result. An agent automates interaction with a large language model (LLM) in AI.
- An assistant is synonymous with an agent. Both terms encompass tools such as OpenAI's GPT Assistants.
- Autonomous agents can make independent decisions, and their distinction from non-autonomous agents is crucial.
- The four main types of LLM interactions include direct user interaction, agent/assistant proxy, agent/assistant, and autonomous agent.
- Multi-agent systems involve agent profiles working together, often controlled by a proxy, to accomplish complex tasks.
- The main components of an agent include the profile/persona, actions, knowledge/memory, reasoning/evaluation, and planning/feedback.
- Agent profiles and personas guide an agent's tasks, responses, and other nuances, often including background and demographics.
- Actions and tools for agents can be manually generated, recalled from memory, or follow predefined plans.
- Agents use knowledge and memory structures to optimize context and minimize token usage via various formats, from documents to embeddings.
- Reasoning and evaluation systems enable agents to think through problems and assess solutions using prompting patterns such as zero-shot, one-shot, and few-shot.
- Planning/feedback components organize tasks to achieve goals using single-path or multipath reasoning and integrating environmental and human feedback.

- The rise of AI agents has introduced a new software development paradigm, shifting from traditional to natural language–based AI interfaces.
- Understanding the progression and interaction of these tools helps develop agent systems, whether single, multiple, or autonomous.