# What is a goal-based agent?

## Goal-based agents, defined

A goal-based agent is an artificial intelligence agent that incorporates a proactive, goal-oriented approach to problem-solving and decision-making. It is an example of agentic AI, in which AI systems actually take actions on behalf of users (as distinct from, say, a simple LLM customer support chatbot).

In the five-level hierarchy of agent complexity, goal-based agents sit squarely in the middle. They are more complex than both simple reflex agents (which follow predefined rules) and model-based reflex agents (which add in an internal model of the world). But they are less complex than both utility-based agents (which can compute tradeoffs using a so-called utility function) and learning agents (which can adapt and improve over time, often through reinforcement learning or deep learning).

Goal-based agents exceed simpler reflex agents by adding a planning function that considers future states — but they stop short of employing the dynamic assessments of more sophisticated agents, instead relying on preprogrammed strategies or decision trees in pursuit of their goals.

Your subscription will be delivered in English. You will find an unsubscribe link in every newsletter. You can manage your subscriptions or unsubscribe here. Refer to our IBM Privacy Statement for more information.

## A real-world use case

One example of a goal-based agent comes from the realm of robotics, specifically warehouse automation. A warehouse robot that needs to pick a certain item from a certain shelf could just react to immediate obstacles, like a robot vacuum bumping against walls (a purely “reactive agent”). More efficient though would be to plan a path that minimizes detours and avoids known obstacles. For instance, in a dynamic environment like a modern warehouse, a planning module can consult its knowledge base, observe the current state and map future states (e.g., by awareness of other bots’ planned motions), all the better to reach its desired outcome efficiently.

## How goal-based agents work

Goal-based agents operate in four stages:

1. Goal definition
2. Planning
3. Action selection
4. Execution

### Goal definition

First, the agent is given a precise definition of success. Unlike utility-based agents, goal-based agents operate under binary, logical conditions. But while a goal-based agent may define success as the flipping of bits from one state to another, there is nothing to prevent such an agent's singular "goal" from being a moderately complex set of propositional and first-order logic. For instance, a robot might set the goal of "for each package currently marked urgent, deliver each from its respective inventory location to the shipping dock." The goal is fundamentally binary (one can either succeed or fail at this goal), but it also has multiple components, allowing complexity.

While it would be nice for a human user to define goals precisely, it is also possible for a human to enter a vaguer objective ("optimize holiday fulfillment") and for an LLM to bridge the gap and define this into a more precise goal or set of goals (for example, "estimate daily throughput of system," "define P packages as highest-priority,"; "set goal to deliver P packages within 24 hours").

### Planning

Having formed its goal, the goal-based agent engages in a bit of planning before moving to execution. For instance, in the warehouse example, the agent will spend some time modeling current and potential conditions in order to choose an optimal path to fulfill the highest-priority orders in time—deciding, for instance, just how often it should make trips from inventory shelves to the shipping bay.

### Action selection

Of course, plans are only as good as the reality they encounter. While some goal-based agents are "conservative" and prefer to stick to plans until they are rendered literally impossible, more flexible approaches come from "opportunistic" agents which, if they encounter impediments to their plans, flexibly calculate a better next action: much like when Google Maps routes around an unexpected traffic jam.

### Execution

In the case of a warehouse robot, sensors on the machine can help monitor the situation in real-time, feeding back crucial data to the planning module. For instance, if a sensor detects that the sub-goal "grasp package" has failed for some reason, a goal-based agent can attempt to diagnose the cause, plan an alternate approach or call for backup.

## When to use goal-based agents versus more complex agent types

The decision of which type of AI agent to use comes down to what type of problem one must solve. So long as they are given specific goals, the AI models underlying goal-based agents can make informed decisions and handle complex tasks—but the criteria for success must be straightforward (often binary).

However, in situations where real-time adaptability is crucial, or where there are multiple goals to optimize among, businesses may want to graduate from goal-based agents to a more sophisticated option, like utility-based agents. A canonical example here would be a autonomous vehicle. When a passenger hails a self-driving car, there are multiple goals to balance: duration, pricing, traffic avoidance, safety. To deliver the best customer experience, an AI-powered vehicle must undergo complex decision-making processes, computing various tradeoffs of possible actions. The resulting utility function will dictate the agent's actions.

## Multi-agent systems

There is nothing to say that an agentic system cannot "mix and match" agent types, with each agent tailored to the complexity of the problem at hand. For instance, one can imagine an example from healthcare: a hospital that employs not just goal-based agents to execute workflows, but also agents of the four other types examined.

At the simplest level, a reflex agent named Vitals Monitor might simply monitor the vital signs of all patients. Its specific objective is to trigger an alarm if a patient's heart rate, say, dips below a certain level—the better to alert a doctor or nurse for human intervention. Such an agent can rely on simple if/then algorithms.

One level up, a model-based reflex agent named Inventory Agent might manage inventory of medicines and supplies for the hospital. It maintains an inner model of current inventory, historical usage patterns and response times of supply chain partners, the better to streamline refill orders.

Third, a higher-level goal-based agent called Discharge Planner might work backwards from the simple binary goal of patient discharge. It would rely on preprogrammed strategies and decision trees, while also considering future states, to coordinate labs, medicines and specialist sign-offs—including all requisite subtasks. If a step is delayed, its planning module can re-run, formulating a new plan. (The goal-based agent, like most of these agents, is likely to be fine-tuned on a large language model.)

Fourth, a utility-based agent called a Bed Assignment Optimizer might assign patients to various rooms, while trying to maximize safety, satisfaction and throughput. Since it must manage multiple goods and complex tradeoffs, the agent works with a utility function, assessing variables like contagiousness, staffing levels, and illness severity.

Fifth and at the highest level, a learning agent called an Intake Assistant employs machine learning, seeking patterns from past experiences in order to improve triage questions, flag high-risk patients and reduce redundant steps. Unlike lower-level agents, this learning agent must continually evaluate evolving data sets, seeking deep patterns that might be invisible to humans.

The five agents work together as a set of virtual assistants to solve complex problems. With the proper orchestration, as well as integration of various capabilities from natural language processing (NLP) and generative AI to computer vision and API tool calls, the multi-agent system is simple where it needs to be simple, complex where it needs to be complex.