Who Decides What Happens Next:
Untangling Agentic Workflows from Autonomous Agents

Adoption of agentic AI is accelerating fast. Deloitte projects that by 2027, as many as half of all companies using generative AI will have launched agentic pilots or proofs of concept. That wave has been big enough to stretch the word "agentic" until it covers almost anything with a model call in it, from a rigid five-step pipeline where one step happens to ask a model for a summary, to a fully self-directing system that plans its own path with no script at all.

01

Those are not the same thing, and treating them as interchangeable leads straight to one of two costly mistakes. You either over-engineer a simple, well-understood task by bolting on autonomy it never needed, or you under-engineer a genuinely open-ended problem by forcing it into a rigid pipeline that shatters the moment reality strays from the plan. The way out of that confusion is to ask a single, clarifying question about any system you are looking at: who decides what happens next, a human who wrote the code in advance, or a model reasoning in the moment?

02

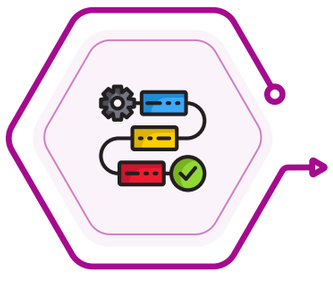
Anthropic draws the foundational line in its widely cited piece on building effective agents. Workflows are systems where models and tools are orchestrated through predefined code paths. Agents are systems where models dynamically direct their own process and tool use, keeping control over how a task gets done. Everything that follows sits underneath that one distinction.

03

Anthropic draws the foundational line in its widely cited piece on building effective agents. Workflows are systems where models and tools are orchestrated through predefined code paths. Agents are systems where models dynamically direct their own process and tool use, keeping control over how a task gets done. Everything that follows sits underneath that one distinction.

The Real Axis Is Predictability, Not Whether AI Is Involved

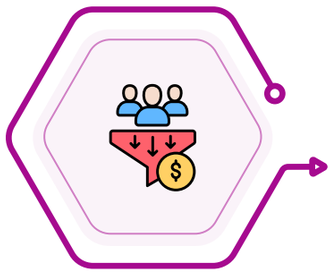
Before comparing architectures, it helps to throw out the wrong question. The question is not "does this system use a model." Almost everything does now. The two questions that actually matter are sharper. First, does this process need to be repeatable, auditable, and explainable step by step? Second, is the correct path even known in advance, or does the system have to discover it at runtime?



A system can lean heavily on a model and still be completely deterministic in structure, a fixed pipeline where one step calls a model for text but the next step is hardcoded no matter what comes back. A system can also be called "agentic" while having almost no real autonomy, a tightly scripted loop with two allowed actions and a hard step limit. The presence of a model call is not the signal. Ownership of control flow is. Google Cloud's design-pattern guidance draws the same operational line: deterministic workflows follow a clearly defined path known in advance, where steps barely change from one run to the next, while dynamic orchestration covers problems where the system itself must work out how to proceed without a predefined script. That spectrum is the territory worth walking through, one stage at a time.

Stage One: Deterministic Workflows

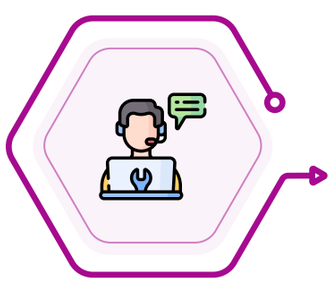
This is the baseline. A deterministic workflow has a known sequence of steps decided at design time by a human, in code. A model can live inside any step, generating text, classifying input, drafting a summary, but it does not choose what happens after its step runs. The orchestrating code does that, regardless of what the model returns.



Picture a customer-message pipeline that extracts the text, classifies it, summarizes it, and sends a notification. A model might label one message "billing" and another "general," yet both messages travel through the exact same four steps in the exact same order. The label is data flowing through a fixed pipe, not a fork in the road. That is the whole definition of deterministic: the route is fixed even when a model is doing real work inside one of the steps.

Stage Two: Orchestrated Workflow

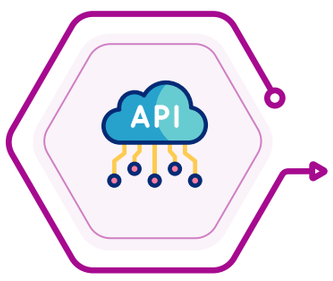
This is the middle ground that gets mislabeled as "agentic" most often, and it is the line most people actually cross when they start using the word loosely. An orchestrated workflow still has a graph of possible paths defined entirely in advance, but which path gets taken now depends on a runtime decision, frequently made by a model.



It is still a workflow. Every branch that could be taken was anticipated and written into the code by a human before the system ever ran. Think of a support router with three handlers, billing, technical, and general, all defined ahead of time. The model reads an incoming message and picks which handler to dispatch to. Different inputs now take different paths, which is genuinely new compared with the deterministic case, but the model is choosing a key off a menu someone else wrote. It cannot invent a fourth handler that was never coded. Google Cloud files this exact behavior under dynamic orchestration, separate from true agents: the system needs to plan and route, but inside a structure a human still fully designed.

Stage Three: Reactive Agents and a Genuinely Open Path

This is where real autonomy starts. The ReAct pattern, short for reasoning plus acting and introduced by Yao and colleagues in 2022, lets the model itself decide at each step what action to take next, based on what it observed from the previous one. There is no pre-written branch covering every case. The agent runs an iterative loop of thought, action, and observation until an exit condition is met, and the sequence itself, how many steps, in what order, using which tools, is not knowable in advance. Only the available actions are fixed. The path through them is not.



The difference is measurable, not rhetorical. Give such an agent a question it can answer from its knowledge base and it might finish in two steps. Give it a question that has no match, and it might take three, choosing on its own to escalate to a human, an action no one hardcoded as the response to that specific situation. The same loop and the same code produce different step counts and different sequences because the model decided the path at runtime from what it observed. That is the concrete meaning of "no predefined code path." In practice, production versions of this pattern keep the running thought-and-observation history in a scratchpad and summarize tool outputs before feeding them back, since dumping raw logs or huge API responses into context tends to confuse the next reasoning step rather than help it.

Stage Four: Autonomous Multi-Agent Systems

The far end of the spectrum builds directly on the ReAct loop, just nested. In a multi-agent setup, an orchestrator runs its own reasoning loop in which some of its available actions are calls to other agents, each running its own complete loop inside. The orchestrator reasons about what to delegate, delegates it, observes the result, and continues, exactly like a single agent, except some of its tools are entire agents rather than simple functions.

Imagine an agent's toolbox where the entries are not search and escalate, but a research agent, a finance agent, and a coding agent, and calling one kicks off that sub-agent's own independent thought-action-observation loop, which might run for several steps before returning anything. Nobody wrote down in advance which sub-agent gets called, in what order, or how many times. Google Cloud labels the most extreme version of this the swarm pattern, a collaborative team of agents with no central orchestrator at all, capable of unusually creative solutions precisely because nothing constrains how they interact. That same lack of structure is the risk: without a human-designed bound on the interaction, a swarm can fall into unproductive loops or simply fail to converge, and the cost of running many agents through many turns compounds quickly. This is where the predictability axis swings hardest in the opposite direction. A deterministic pipeline gives you the same output structure every time by construction; a swarm gives you the flexibility to handle a problem nobody anticipated, at the price of not being able to predict what it will do or how long it will take.

Why the Distinction Decides What Teams Actually Ship

This is not an academic exercise. It has a direct, measurable effect on production. For all the noise around autonomous agents, it was AI workflows, not fully autonomous agents, that won the production battle in 2025. Workflows remain the dominant pattern behind successful generative AI deployments, while fully autonomous multi-agent systems stay largely exploratory outside narrow domains.

01

The reason maps straight back to the predictability axis. Agentic systems are non-deterministic by nature, so identical inputs can produce different outputs across separate runs, which is a serious liability in regulated, auditable, or otherwise high-stakes processes. If a process has to be explainable step by step to a compliance team or a regulator, it is not agent territory by default. It needs guardrails and human-in-the-loop checkpoints layered on before it can be trusted with real consequences.

02

The pattern emerging in mature systems is hybrid, not a pick-one decision. A higher-level agent sets goals and orchestrates the overall task, while critical, well-understood computations still run inside deterministic modules a human has fully specified. A medical diagnostics system, for instance, might use an agent to interpret ambiguous symptoms and decide which tests to order, genuine autonomy, because the right sequence of tests cannot be known in advance, while each test itself runs through a validated, deterministic pipeline, because that part has a known correct path and no reason to introduce variability.

The Takeaway

"Agentic workflow" and "autonomous agent" describe two ends of one spectrum, not two rival technologies. The four stages here, deterministic, orchestrated, reactive, and autonomous multi-agent, are not a ranking from worse to better. They are different answers to the same question: who decides what happens next, and was that decision made by a human writing code in advance or by a model reasoning at runtime?

Deterministic workflows hand you auditability and repeatability by construction; the same input takes the same path every time. Reactive and multi-agent systems trade that guarantee the ability to handle problems whose shape genuinely cannot be anticipated. Neither property is free, and neither architecture is correct by default. The systems that hold up in production do not pick one extreme and apply it everywhere. They place each piece of the problem at the point on the spectrum it actually calls for, fixed structure wherever a known correct path exists and repeatability matters, and real autonomy reserved only for the parts that have no predefined correct path to follow in the first place.