

Build, buy or agent: deciding how AI work gets done

Every piece of AI work gets done by building it, buying it or deploying an agent, and the choice is often made by habit or fashion. This paper argues for choosing by the properties of the task and the organization, and for writing the choice down.

Author	Amit Govil, Founder and CEO, XONIK
Published	October 22, 2026
Reading time	23 min read
Online	https://xonik.com/insights/white-papers/build-buy-or-agent/

IN BRIEF

AI work should be built, bought or given to an agent according to the task, not fashion: how variable it is, what a wrong answer costs and how soon it is caught, who must own it, what it touches and how often it changes. This paper sets out what each route commits you to and a method for choosing.

KEY FINDINGS

1. What a wrong result costs, and how soon it is caught, comes before everything else

The earliest properties can rule a route out before the others are weighed. Where the answers conflict, as when a task is variable but a wrong result is costly, the cost of error decides, because it sets the evidence the work needs before it runs.

2. Each route fails where it is weakest, and the root is the same every time

A build fails by neglect, a product by fit and an agent by drift. In each case nobody was named to prevent the failure.

3. Each route is a commitment that lasts as long as the task, and it is weakest when the owner moves on

Ownership, change, evidence and exit differ by route. The reasons for the choice leave with the owner, so a decision record kept from the first day is the cheapest protection.

4. Buying a product or deploying an agent moves the work, not the accountability

NIST expects roles, executive responsibility and third-party risk to be handled inside the buyer's own organization. Under the EU AI Act, a change of name, of substance or of purpose can make a deployer the provider of a high-risk system, which is a question for counsel.

5. An agent is not a way of avoiding the work of building

It replaces a procedure with a goal and lets the model choose the plan and the tools. The work moves into tool definitions, permissions, evaluation and monitoring, which is why the route suits tasks where a wrong action is cheap to catch and to undo.

6. The record of the decision matters more than the choice

It holds the task as one sentence, the answers to the properties, the route, the owner, the conditions under which the choice would change and a review date. A decision with its reasons can be revisited calmly.

What is the choice, and why does it matter now?

Building means that the organization writes software, rules and model calls that it owns, runs and changes. Buying means subscribing to a product that does the task, which the organization configures and the vendor maintains. Deploying an agent means giving a model a goal and a set of tools and letting it decide the steps, within limits that are written down. One organization will use all of them, for different tasks, so the question is which task goes where.

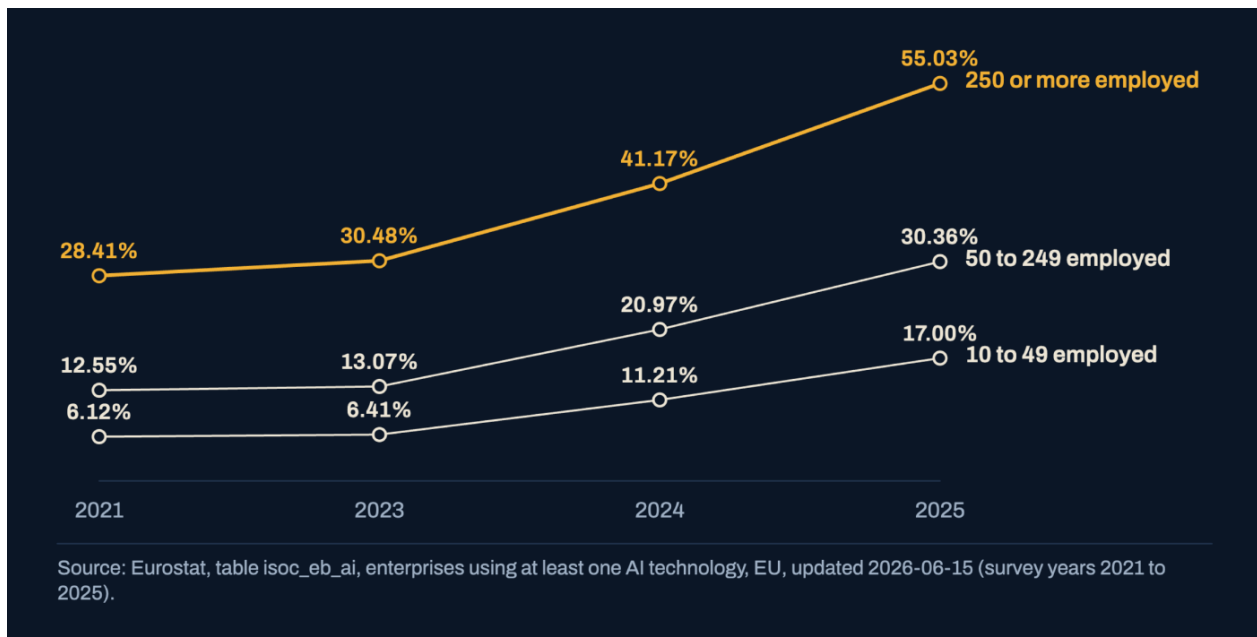
The line between the routes is not whether a model is inside. Anthropic's engineering article on building effective agents separates workflows, where models and tools are orchestrated through predefined code paths, from agents, where the model dynamically directs its own process and tool usage. By that line, a system that calls a model inside steps you wrote is a build, or a product, and not an agent. Who decides the path, your code or the model, shapes control, risk and accountability.

The argument of this paper is that the route follows from the task and the organization, taken in a fixed order. What a wrong result costs and how soon it is caught comes first. How variable the task is comes second. Then who must own the result, what it must touch, how often it changes, and what you can inspect and switch off. Fashion enters none of these. A route is also a set of commitments that last as long as the task does, which is why the choice deserves a written decision with an owner.

The question is being met by a widening set of organizations. Exhibit 1 charts Eurostat's measure of EU enterprises that use at least one of the AI technologies it lists, by size class, for the survey years 2021 to 2025. The share rose in every size class shown, and the larger enterprises are well ahead of the smaller. The measure does not identify agents, does not say whether the technology was built or bought, and does not say how well it works. It supports a narrow point: more organizations reach this decision with each survey year, and they reach it unevenly.

EXHIBIT 1

Enterprises using AI, by size class, EU



Adoption of AI technologies is rising in every size class and unevenly, so a growing set of organizations meets the build, buy or agent decision.

Method and limits

This paper is an argument, and it rests on a stated base. One part is what the firm has seen in delivery since 2006, set out as mechanism and never as a case, count or share. Another is public primary documents, each read on 20 October 2026 for the passages cited: the EU AI Act, Regulation (EU) 2024/1689, in the Official Journal text, with Regulation (EU) 2026/1744 that amends it; the NIST AI Risk Management Framework (NIST AI 100-1, January 2023); the OWASP Top 10 for LLM Applications 2025; OWASP's announcement of its Top 10 for Agentic Applications; and Anthropic's engineering article "Building effective agents". The last is one published statistic, charted from a stored copy of Eurostat's table.

Anthropic builds agent technology, so its article is a vendor describing its own experience, and it is cited only for when agents are warranted and what they cost. This paper recommends no vendor and no product.

- It is an argument and not a survey. It does not say how often any route succeeds or fails, or which is cheaper in general, because cost depends on the case.
- The EU AI Act is cited for what its articles say about roles. The duties discussed apply to systems classed as high-risk, from dates the 2026 amendment sets. Whether a system is high-risk, and whether a change makes an organization a provider, is a question for counsel. These readings are not a legal opinion.
- The full OWASP Top 10 for Agentic Applications sits behind a download that could not be opened. It is described only as far as OWASP's announcement and project page describe it.
- Anthropic's article notes that much of the tooling it describes has changed since it was written. We rely on its design principles and not on its tooling.

- The decision method is a working test drawn from practice, not a validated instrument.
- No client work appears here. Our engagements are under NDA.

Which properties of the task decide the route?

Habit often decides these choices: the team that builds builds, the team that buys buys, and the new thing becomes an agent. A better basis is a set of properties, taken in an order, because the earliest of them can rule a route out before the others are weighed.

The first is what a wrong result costs and how soon it is caught. A wrong answer that a person sees within the hour and can reverse is a different problem from one that is acted on, copied into other records and found at month end. The second sets how much evidence the work needs before it goes live. NIST MAP 3.5 asks that processes for human oversight are defined, assessed and documented.

The second is how variable the task is. If the inputs and the right output are known, software you can read and test does the work without variation, whether you write it or buy it. The case for a model-directed route grows with variability. Anthropic's article puts the line there: workflows offer predictability and consistency for well-defined tasks, while agents suit open-ended problems where the number of steps is hard or impossible to predict. It advises finding the simplest solution possible, and notes that agentic systems often trade latency and cost for better task performance.

The third is who must own the result. Someone in the organization answers for the output whichever route runs it. Ownership means the authority to change the behavior, to see how an answer was produced and to stop it. A route that denies the owner a view into the answer is a risk to be weighed, not a convenience.

The fourth is what data and systems the work must touch. A task that reads one document and returns a draft needs little access. A task that reads customer records, updates a ledger and sends messages needs credentials in more than one system, and each credential is an authority that someone must grant, scope and revoke.

The fifth is how often the work changes. A requirement that changes monthly fits badly with a product on a fixed roadmap, and fits a build only while the people who can change it remain. A change to an agent's instructions is quick to make and hard to test, because the same instruction can produce a different path tomorrow.

The sixth is what you can inspect and switch off. In a build you can usually read the logic, if logging was designed in. In a product you see what the vendor exposes. In an agent you see only what its planning steps and tool calls record. Anthropic lists showing the agent's planning steps among its principles, and describes stopping conditions and checkpoints for human feedback. NIST MANAGE 2.4 asks that someone is assigned, and a mechanism exists, to take a system out of use when its results depart from its intended use.

Each route invites its own failure where it is weakest on these properties. A build fails by neglect: the team moves on, and the system keeps running with nobody able to change it. A product fails by fit: it does much of the job in a demonstration, and the remainder is done by hand, outside the product. An agent fails by drift: its remit grows because that is convenient, nobody rewrites the instructions, and its behavior changes when the model underneath changes. The root is the same each time. Nobody was named to prevent the failure.

WHERE IT GOES WRONG

Each route fails where it is weakest

A build fails by neglect: the team moves on, and the system keeps running with nobody able to change it. A product fails by fit: it does much of the job in a demonstration, and the remainder is done by hand. An agent fails by drift: its remit grows because that is convenient, and nobody rewrites the instructions.

What does each route commit you to over time?

The choice is made once and lived with for years, so it helps to ask what each route commits you to on ownership, change, evidence and exit. Exhibit 2 sets the routes side by side, and Exhibit 3 shows when the commitments fall due.

Ownership differs in what you carry. In a build you own the code and the knowledge of how it works, and you carry its upkeep in people. In a product the vendor owns the product, and you own your configuration, your data and the process around it. With an agent you own the remit, the tool definitions and the test cases, while a model provider you do not control can change the model beneath them.

Change differs in who sets the pace. A build changes at the speed your own team can manage, a product on the vendor's release schedule and contract. An agent changes by editing instructions, which is quick, but every edit and every model update needs the work checked again.

Evidence differs in what you can show an auditor or a regulator. For a build it is your own tests and logs. For a product it is the vendor's documentation together with your own checks on your own examples. For an agent it is your evaluation of the results and a record of each step and tool call. Anthropic's article says that measuring performance and iterating is the key to success, and recommends extensive testing of agents in sandboxed environments.

Exit differs in what has to come out. A build leaves code and data in your hands, though the knowledge of why it works as it does may be the real lock-in. A product needs your data to come out in a usable form and a substitute to exist. An agent's remit is written down and can be moved, but accumulated memory or instructions tuned to one model carry a cost. NIST GOVERN 1.7 asks for processes to decommission AI systems safely, and GOVERN 6.2 for contingency processes when third-party AI systems fail.

EXHIBIT 2

What each route commits you to

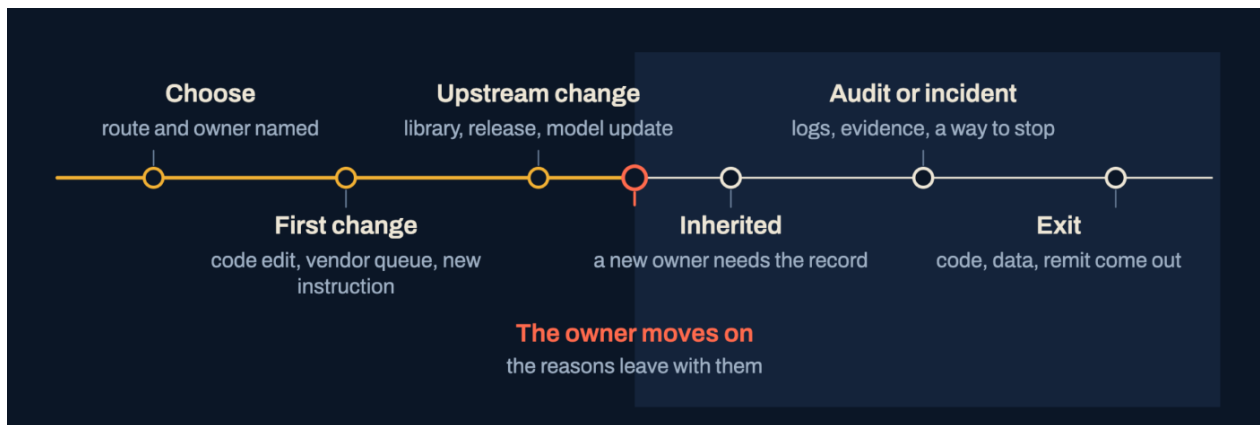
Commitment	Build	Buy	Agent
You own	Code, tests and the knowledge of how it works	Configuration, data and the process around the product	The remit, the tool definitions and the test cases
Change happens	At your team's pace and cost	On the vendor's release schedule and contract	By editing instructions, then checking again after each edit and each model update
Evidence you hold	Your own tests and logs	Vendor documentation and your checks on your cases	Your evaluation and a record of each step and tool call
Exit means	Moving code and data, and the knowledge with them	Getting data out in a usable form and finding a substitute	Moving the remit, and unpicking memory and model-specific tuning

Where it fails first	Neglect, when the builders move on	Fit, when the remainder is done by hand	Drift, when the remit grows and the model changes
----------------------	------------------------------------	---	---

The routes differ less in what they do than in what they leave with you: who owns the result, who sets the pace of change, what evidence you hold and what has to come out when you leave.

EXHIBIT 3

When the commitments fall due



The commitments that fall due late, evidence and exit, depend on a record made early, which is lost when the owner moves on without leaving one.

How do the commitments play out in time?

At the start every route looks alike, because the owner is in the room. The first change request shows the route's real shape: an edit, a place in a queue or a new instruction. An upstream change then arrives, whether a library, a release or a model update, and each route needs someone to read it and test again.

The commitments are at their weakest when the owner moves on. The reasons for the choice leave with that person, and the successor inherits a system without a record of what it was meant to do or when to reconsider it. The audit or the incident arrives when the system is least understood, and exit is the last commitment and the first one skipped in design. A decision record kept from the first day is the cheapest protection against each of these.

Buying a product or deploying an agent moves the work onto someone else's system, not the answer.

Where does accountability sit when you buy a product or deploy an agent?

Buying a product or deploying an agent moves work onto someone else's system. It does not move the answer. The NIST framework says that actors across the lifecycle see different risks: an organization developing an AI system often will not know how it may be used (section 1.2.4), and a developer of a pre-trained model can have a different risk perspective than the organization deploying it, which may not recognize risks the developer did not perceive (section 1.2.1).

NIST turns this into expectations inside the buyer's own organization: roles and responsibilities for AI risk are documented and clear (GOVERN 2.1), executive leadership takes responsibility for decisions about AI risk (GOVERN 2.3), and third-party risks are covered by policy (GOVERN 6.1) and monitored regularly (MANAGE 3.1). Each happens in the buyer's organization and cannot be delegated to the vendor.

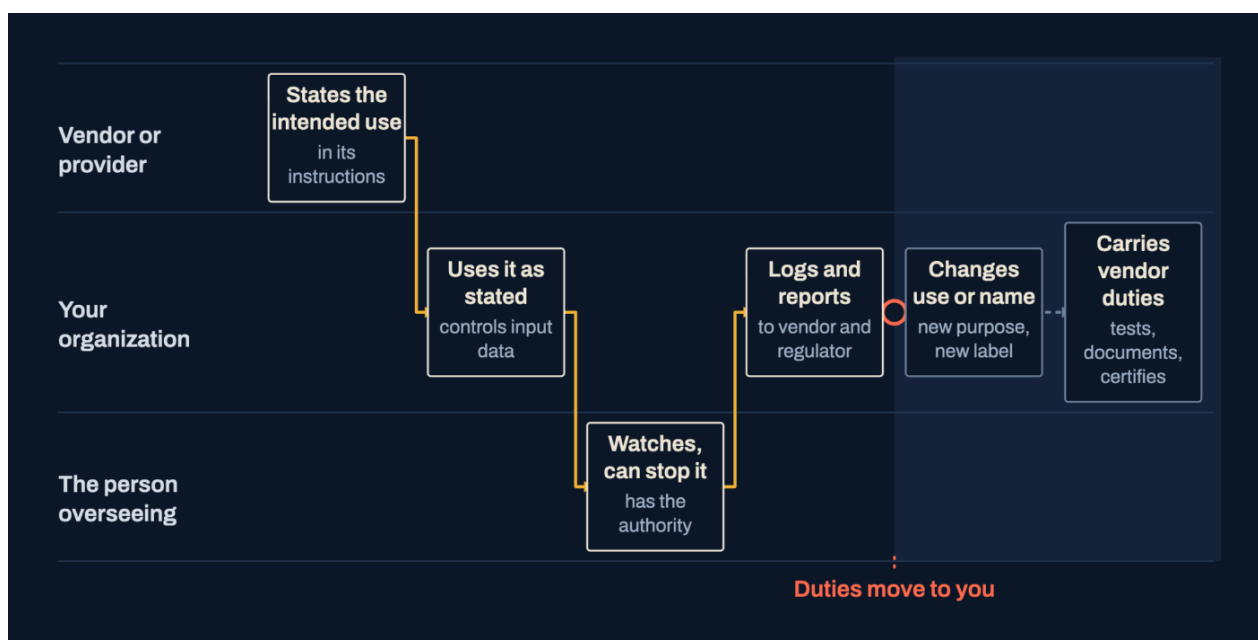
The EU AI Act gives the same idea legal form through the roles of provider and deployer. A provider, under Article 3(3), is whoever develops an AI system, or has one developed, and offers it or starts using it under its own name or trademark. A deployer, in Article 3(4), uses an AI system under its own authority. Buying a product usually makes an organization a deployer. Building a system for its own use generally makes it a provider, because "putting into service" in Article 3(11) includes supply for own use.

The Act's heavy duties attach to systems it classes as high-risk, through Article 6 and Annex III, which the 2026 amendment applies from 2 December 2027 for Annex III systems. A business use outside those areas does not carry these duties, though other law may. For a deployer of a high-risk system, Article 26 requires use in line with the instructions for use, human oversight given to people who have the competence, training and authority for it, and the support, monitoring with a duty to inform the provider and to suspend use where a risk appears, and retention of the logs it controls for at least six months. Article 14(4) requires that those overseeing can understand the system's limits, stay aware of automation bias, override its output and stop it.

These are the properties of the previous section written as law, and they show what a buyer needs from a vendor before signing: the instructions for use, the logs, a route for reporting a fault, and a way to stop the system. Article 25(4) points the same way: where a third party supplies tools, services or components to the provider of a high-risk system, a written agreement settles what information, access and help the provider receives. Accountability is written in the contract and checked before purchase.

EXHIBIT 4

How responsibility moves



Responsibility passes from vendor to organization to the overseer and back, and it moves to the organization itself when the use or the name changes.

When does a deployer become a provider?

Article 25(1) names the cases. A deployer, distributor, importer or other third party is considered the provider of a high-risk AI system, with the provider's duties under Article 16, if it puts its name or trademark on a high-risk system already on the market, without prejudice to contractual arrangements that allocate the obligations otherwise; if it makes a substantial modification to a high-risk system that remains high-risk; or if it modifies the intended purpose of an AI system, including a general-purpose AI system, that was not high-risk, so that it becomes high-risk. Article 3(12) defines the intended purpose as the use the provider intended, as stated in the instructions for use, the promotional or sales materials and statements, and the technical documentation.

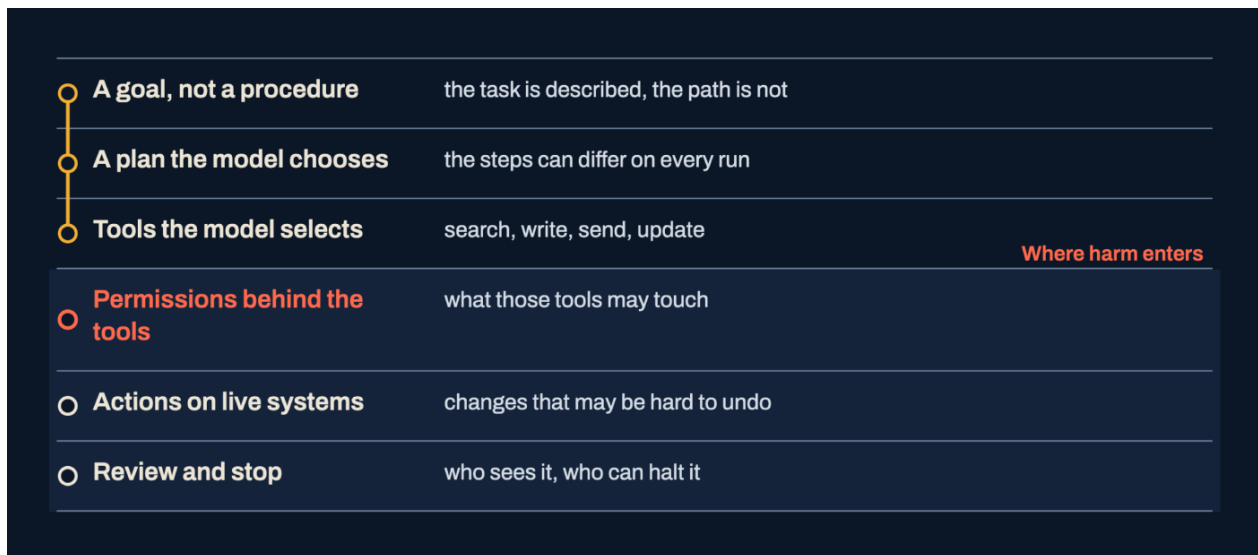
Exhibit 4 shows why this matters for the choice. An organization that buys a product and uses it as stated is a deployer. One that relabels it, changes it substantially, or points a general-purpose system at a purpose that makes it high-risk may become a provider, and then Article 16 applies: a quality management system, documentation, logs, a conformity assessment, a declaration of conformity, registration and corrective action. The initial provider stops being the provider and must cooperate with the new one. As amended in 2026, Article 25(2) has it supply technical documentation, known limitations and failure modes, and targeted technical access, unless it clearly specified that its system was not to become high-risk. An agent assembled from a general-purpose model and aimed at an Annex III area is the case the third limb describes. Whether it applies is a legal judgment, but the practical rule is plain: a change of purpose is a decision with an owner, and it is recorded.

What does an agent add that a build or a product does not?

A build or a product, even one that calls a model, follows a path that someone decided in advance. In Anthropic's terms it is a workflow. An agent is a system in which the model directs its own process and tool usage, and keeps control over how it accomplishes the task. Exhibit 5 shows the layers. The upper layers are what the agent adds: a goal in place of a procedure, a plan the model chooses, tools it selects. The lower layers are what that choice puts at stake: the permissions behind each tool, the actions taken on live systems, and the review that can stop them.

EXHIBIT 5

What an agent adds to a task



What an agent adds is choice at the top of the stack, and the risk sits below it, in the authority that the model's choices are exercised through.

What do vendor and security guidance say about the risks?

Anthropic's article is direct about the cost. It says that the autonomy of agents means higher costs and the potential for compounding errors, and it recommends extensive testing in sandboxed environments along with appropriate guardrails. It finds agents best for work with clear success criteria, feedback loops and meaningful human oversight, and says that even for coding agents, whose results automated tests verify, human review remains crucial. It also suggests starting with direct calls to a model's interface, because frameworks can obscure the underlying prompts and responses and make an agent harder to debug.

OWASP's Top 10 for LLM Applications 2025 names the risk that follows. Its entry LLM06, Excessive Agency, is the vulnerability that enables damaging actions in response to unexpected, ambiguous or manipulated outputs from a model, whatever causes the malfunction. Its root causes are excessive functionality, excessive permissions and excessive autonomy. Its examples are ordinary: a tool meant to read documents that can also delete them, or one that reads data through an identity able to update it. Its mitigations read as a design brief: limit tools and their functions to what is necessary, avoid open-ended tools such as one that runs any shell command, limit permissions, act in the user's own context, require a person to approve high-impact actions, and enforce authorization in downstream systems instead of relying on the model to decide what is allowed. Logging, monitoring and rate limiting are listed as limiting the damage, not preventing it.

Entry LLM01, Prompt Injection, explains how an agent comes to be misdirected. Indirect injection arises when the model accepts input from external sources such as websites or files. OWASP says that severity depends on the agency with which the model is architected, that retrieval and fine-tuning do not fully mitigate it, and that it is unclear whether fool-proof prevention exists. For an agent that holds tools, an instruction hidden in a page can reach them, and limiting what the tools may do is the defense that remains.

OWASP's announcement of its Top 10 for Agentic Applications, on 9 December 2025, lists risks that include agent goal hijack (ASI01), tool misuse (ASI02), identity and privilege abuse (ASI03), memory and context poisoning (ASI06), cascading failures (ASI08) and human-agent trust exploitation (ASI09). It describes confident, polished explanations misleading human operators into approving harmful actions. That matters for the choice, because the safeguard the method leans on, a person reviewing, can fail when reviewers over-trust fluent output. Article 14(4) of the EU AI Act names the same tendency, automation bias.

What an agent adds, then, is a requirement to design each protection: tools minimized, permissions scoped to the task and the user, high-impact actions approved by a person, every step logged, a way to stop. These are build-time activities. An agent is not a way of avoiding the work of building. It moves the work into tool definitions, permissions, evaluation and monitoring, which is why the agent route suits tasks where a wrong action is cheap to catch and to undo.

What changes the outcome?

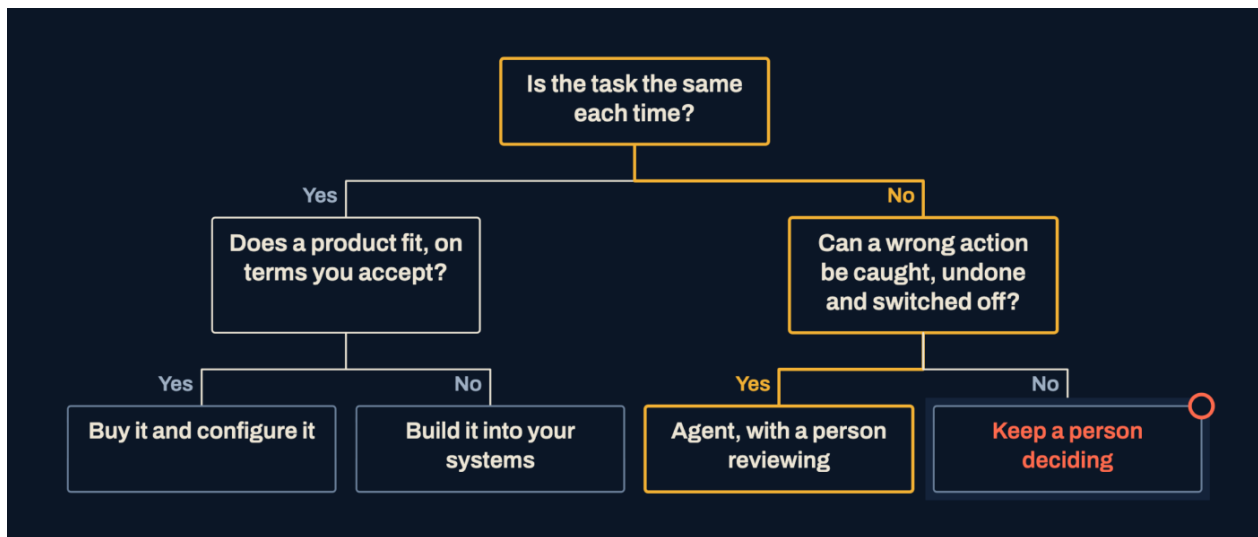
Practice changes it. The first practice is to decide per task, not per organization or per vendor. A policy of agents first or buying first is a fashion with a budget behind it.

The second practice is to take the properties in order. Exhibit 6 sets the order as a tree. The first question is whether the task is the same each time, because a task that is the same each time has no need of a model to choose the path. If it is, the question is whether a product fits on terms you accept, and the answer is a purchase or a build. If it is not, the question is whether a wrong action can be caught, undone and switched off. Where it can, an agent with a person reviewing is justified. Where it cannot, a person keeps deciding and software assists, and the task can be reconsidered once it is narrowed.

The tree is a first pass and not a verdict. Ownership, data and exit act as gates on every outcome. A product whose terms keep your data out of reach fails the question about terms. A build with no named maintainer fails at ownership. An agent whose tools cannot be narrowed fails at the last question. Where answers conflict, as when a task is variable but a wrong result is costly, the cost of error decides, because it sets the evidence the work needs before it runs.

EXHIBIT 6

How to choose a route for one task



The route follows from the task: sameness first, then fit, then whether errors can be caught, undone and stopped.

What is the third practice?

The third practice is to write the decision down with its reasons: the task as one sentence, the answers to the properties, the route chosen, the owner, the conditions under which the choice would change, such as a vendor's terms changing or an agent's results in your own checks, and a date for review. The record matters more than the choice. A decision with its reasons can be revisited calmly, and a decision without them is reopened each time someone new asks why.

How do you apply the method to one task?

1. State the task in one sentence

The sponsor writes it with the person who does the work today: what goes in, what comes out, who uses the result. What exists at the end is that sentence, agreed by both.

2. Answer the properties for this task

The sponsor puts them to the people who carry the consequences of an error, to the technology lead and to risk, and not only to the sponsor. Disagreements are noted. What exists at the end is one page of answers on the cost and detection of error, variability, ownership, access, change, and what can be inspected and stopped.

3. Run the tree and name the gates

The technology lead takes the answers through the tree. What exists at the end is a provisional route and the gates it must pass: a named owner, terms on data that you accept, and a way out.

4. Test the route on your own cases

The owner and the team collect real cases with known right answers and run the route on them: a product on a trial, a build as a prototype, an agent in a sandbox. What exists at the end is a record of results, with errors sorted by kind and by how each was caught.

5. Fix the boundaries

For a product, the contract states data use, notice of changes, access to logs and export of data. For a build, a maintainer and a run book are named. For an agent, the tools, permissions, approvals, step records and stop are written down. The owner, security and legal do this together. What exists at the end is a boundary document the owner signs.

6. Settle where accountability sits

The executive owner and counsel decide whether any law or regulator treats the organization as provider or deployer of this system, and which changes of purpose or name would alter that. What exists at the end is a note of the answer and the changes that would reopen it.

7. Record, run and review

The decision owner writes the record described above, with the exit plan, and sets the review date. What exists at the end is a decision that can be revisited by someone who was not in the room.

What should a leader ask of a proposal?

- The task can be stated in one sentence that a stranger would understand, including what it produces and who uses it.
- We know what a wrong result costs and how soon someone would see it.
- We know whether the task is the same each time, and where it varies.
- A named person owns the result and can change it, see how an answer was produced and stop it.
- Every system and kind of data the route must touch is listed, with who grants each credential.
- For a bought product, the contract covers our data, notice of changes, access to logs and export of our data in a usable form.
- For a build, someone other than the original author can change it, and a run book exists.
- For an agent, its tools are the smallest set the task needs, its permissions are scoped to the task and the user, and a person approves any action that is hard to undo.
- We have run the route on our own cases and kept the results.
- We know whether any law or regulator treats us as provider or deployer of this system.
- We can switch it off without stopping the business, and we have tried it.
- The decision record holds our reasons, the conditions for changing the choice, an owner and a review date.

What do readers ask before choosing a route for AI work?

How do I decide whether a piece of AI work should be built, bought or given to an agent?

Decide per task, not per organization or per vendor. Take the properties in order: what a wrong result costs and how soon it is caught, how variable the task is, who must own the result, what it touches, how often it changes, and what you can inspect and switch off..

If we buy a product or deploy an agent, does accountability move to the vendor?

No. Buying or deploying moves the work, not the answer. NIST expects roles, executive responsibility and third-party risk to be handled in your own organization. Under the EU AI Act, a change of name, substance or purpose can make a deployer the provider, a question for counsel.

When is an agent the right choice?

When a wrong action is cheap to catch and to undo. An agent replaces a procedure with a goal and lets the model choose the plan and tools, so the work moves into tool definitions, permissions, evaluation and monitoring. Where a wrong action cannot be caught, a person keeps deciding.

What should we write down after we have chosen?

A decision record: the task as a sentence, the answers to the properties, the route chosen, the owner, the conditions under which the choice would change, and a review date. The record matters more than the choice, because a decision with its reasons can be revisited calmly.

Sources and notes

Every document was read on 20 October 2026 for the passages cited. The EU texts were read in the Official Journal versions served by the Publications Office of the European Union. The OWASP Top 10 for Agentic Applications itself was not read, only OWASP's announcement and project page.

1. [Regulation \(EU\) 2024/1689, the AI Act: Articles 3, 14, 16, 25 and 26](#)
2. [Regulation \(EU\) 2026/1744, amending Regulation \(EU\) 2024/1689 \(Digital Omnibus on AI\): Articles 25 and 113 as amended, 8 July 2026](#)
3. [Artificial Intelligence Risk Management Framework 1.0 \(NIST AI 100-1\), NIST, January 2023. NIST says a revision is under way.](#)
4. [OWASP Top 10 for LLM Applications 2025, LLM06 Excessive Agency](#)
5. [OWASP Top 10 for LLM Applications 2025, LLM01 Prompt Injection](#)
6. [OWASP GenAI Security Project, Top 10 for Agentic Applications, announcement of 9 December 2025](#)
7. [OWASP GenAI Security Project, OWASP Top 10 for Agentic Applications for 2026 \(project page; the full document was not read\)](#)
8. [Anthropic Engineering, Building effective agents, 19 December 2024](#)
9. [Eurostat, table isoc_eb_ai, artificial intelligence by size class of enterprise, updated 15 June 2026](#)

USE THIS

[Run your decision through the decision tool](#)

[Build the business case with the spreadsheet](#)

Take one decision you made recently by habit and run it through the tree in this paper. Where the answers disagree with what you chose, you have found a decision worth reviewing.

This document was made from the article at <https://xonik.com/insights/white-papers/build-buy-or-agent/>. The online article is the current version.