

Governing agent decisions

An operating model for ownership, release conditions, exceptions and accountable change.

Governance becomes useful when it changes a decision. Connect each agent workflow to an accountable owner, explicit operating conditions, proportionate evidence and a practical route for exceptions and withdrawal.

FOR Business sponsors, AI governance leaders and delivery executives

Softlander Consultancy FZE LLC | 2026-09-16 | v1.0

The operating model and scenario are illustrative. They do not constitute legal advice, certification or evidence of a client implementation.

Make governance answer a decision

A register of AI applications can tell an organization what it knows about. It cannot, on its own, tell a team whether an agent may issue a refund, approve a supplier, disclose a document or continue operating after a material change. Those questions require decisions with owners, evidence and boundaries. Governance earns its place when it makes those decisions clearer and more consistent.

The central recommendation is to govern a workflow's decision rights, rather than treating an "AI project" as one undifferentiated object. Record what the system recommends, what it executes, who may override it, which conditions make its use acceptable and what events reopen the decision. A product can contain several workflows with different consequences. A single organization-wide label can hide that variation.

NIST AI RMF 1.0 organizes risk management around Govern, Map, Measure and Manage, with governance spanning the other functions. [1] The operating model developed here translates that broad structure into a small set of recurring business decisions. It is a practical design proposal, not a claim that completing a form implements the framework.

Consider a customer-service agent that drafts explanations, proposes compensation and closes support cases. Drafting, compensation and closure require different owners and evidence. A draft can be reviewed before use. Compensation creates a financial consequence. Closing a case can prevent a customer's issue from receiving further attention. Governance should make these differences visible without forcing every routine draft through an executive committee.

The model has four connected records: a workflow charter, a release decision, an exception record and a change decision. They should link to existing delivery and risk systems instead of duplicating them. A small team may maintain them together; a large enterprise may distribute them across tools. Their value comes from the decisions they preserve, not the software used to store them.

Define the boundary that an owner can accept

A useful workflow charter begins with the business activity and affected people. "Improve service productivity with AI" is too broad to govern. "Prepare responses to account-access questions for an adviser to review" establishes a purpose, a user and an action boundary. It can then specify excluded requests, data sources, external tools and the point at which a human takes responsibility.

Name a business owner who can accept the operational consequences, a technical owner who can implement controls and an operational owner who can stop or constrain use. These may be different people. Avoid appointing a committee as the only owner: committees can advise and authorize within a mandate, but someone must respond when a live process leaves its agreed boundary.

Also identify affected stakeholders who are not the purchasing team. Employees may need a route to challenge an automated work allocation. Customers may need a correction path when an agent misstates their account status. A supplier may be affected by an inaccurate classification. These perspectives help reveal consequences that a feature backlog may not capture.

The OECD AI Principles include accountability, transparency, robustness and respect for human rights and democratic values. [2] Those principles motivate concrete questions here: who can explain a decision, who can correct it, and who bears the practical burden when it fails? They do not establish a universal legal rule for every deployment.

Describe the fallback process before approving the automated one. If the agent is unavailable or restricted, can work return to a human queue, a simpler rules-based process or a read-only mode? Name the owner and capacity assumptions for that fallback. A manual alternative that nobody is staffed or authorized to operate is an intention, not an operational control.

Finally, record the exclusions that matter to the business. An agent may support general service inquiries while excluding vulnerability disclosures, legal disputes or decisions about vulnerable customers. Such exclusions should appear in routing and escalation behavior, not merely in a policy document. The charter becomes useful when a delivery team can turn it into test cases and an operator can recognize a breach.

A charter describes a governable workflow

Purpose

Business activity
Affected people
Expected benefit

Accountability

Business owner
Technical owner
Operational owner

Boundary

Permitted actions
Excluded situations
Usable fallback

A product name is too broad: govern the actions and consequences within it.

Figure 1. Purpose, ownership and boundaries form a workflow charter that delivery teams can use. Original Softlander conceptual diagram, 2026-09-16.

Allocate decision rights before creating committees

Different decisions need different participants. The sponsor chooses whether the business problem warrants investment. The workflow owner decides whether the operating process is acceptable. Technical reviewers assess evidence within their competence. Security, privacy, legal or domain specialists address issues that need their judgment. The release authority considers the combined evidence and remaining uncertainty under an agreed mandate.

These roles should not be confused with a promise of independent assurance. A person who designed a control can provide valuable technical explanation, but that is not the same as an independent opinion about it. Where separation is important, record who performed the work, who reviewed it, what relationships may affect judgment and which decision still belongs to the client organization.

A compact decision-rights table is often more useful than a large responsibility matrix. For each decision, identify one accountable role, required contributors, the evidence needed and the action if agreement is not reached. Examples include accepting a data source, expanding an agent's authority, releasing a material change and accepting an unresolved risk for a limited period.

Disagreement needs a route, not a forced average. If security considers a proposed connector unacceptable while the business owner wants the feature, preserve both positions and escalate through the organization's existing authority. Do not turn incompatible judgments into a numerical score that conceals the contested assumption. The eventual decision should state whose authority permits it and what conditions limit it.

Some teams fear that formal decision rights will slow delivery. That concern is reasonable when governance requires repeated approval of unchanged, low-consequence work. A better design pre-authorizes bounded changes that meet defined conditions and escalates only changes that cross them. For example, correcting a spelling error in a user interface can follow routine release controls; adding a tool that sends external messages should reopen authority and disclosure questions.

NIST SP 800-53 includes configuration management and assessment-related controls alongside security and privacy controls. [3] Use existing change and risk processes as the foundation where they already work. Agent governance should add the missing behavioral and decision context, not create a parallel bureaucracy that ignores established operating responsibilities.

Require evidence proportional to the consequence

A release decision should say what is being allowed, for whom, with which controls and on what evidence. The evidence should address the actual consequence. An agent that summarizes internal guidance may require source fidelity and access checks. An agent that changes customer records additionally requires authorization, transaction integrity, recovery and escalation evidence.

Avoid asking for every possible artifact on every project. Start with the most important failure scenarios and work backwards to the evidence that would affect the decision. A privacy reviewer may need the actual data flow and retention configuration, not a general statement that the platform is secure. An operational owner may need a demonstrated fallback and a runbook, not a benchmark score for the underlying model.

The NIST Generative AI Profile identifies risks including confabulation, data privacy and human-AI configuration, and gives lifecycle risk-management considerations. [4] These categories are useful prompts for finding omitted questions. They are not a substitute for showing how a specific failure would affect this workflow and which control addresses it.

Distinguish evidence that a control exists from evidence that it works under the relevant conditions. A screenshot of an approval button shows an interface. A test that changes the transaction after approval and confirms that execution is blocked addresses a more substantive property. Both may belong in the record, but they should not be assigned the same meaning.

The release record should preserve unresolved issues, including those accepted for a bounded deployment. Specify the restriction, owner, review trigger and response if the assumption proves false. A limited internal rollout with human review may be defensible when autonomous external execution is not. The important distinction is that the restriction is part of the decision and operating configuration, not an informal hope.

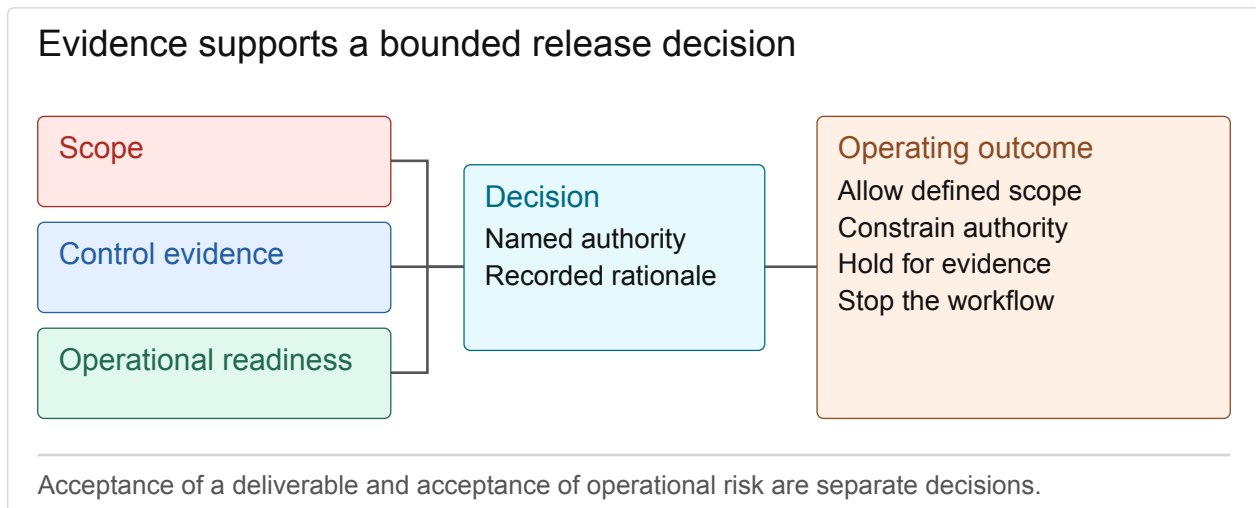


Figure 2. A release record connects evidence to a specific operating outcome and its conditions. Original Softlander conceptual diagram, 2026-09-16.

Acceptance of a deliverable is also different from acceptance of operational risk. A client can accept a report because it meets the agreed scope while choosing not to release the system it examines. Keep those decisions separate in procurement and delivery records. Otherwise, signing off an engagement can accidentally be interpreted as approving the agent's behavior.

Manage exceptions as expiring decisions

Exceptions are inevitable when dependencies, evidence and commercial deadlines do not align. An exception process is useful when it makes a departure visible, bounded and revisitable. It becomes harmful when it turns unresolved issues into a permanent list that nobody owns.

An exception should identify the rule or condition being departed from, the reason, the affected workflow and the practical exposure. It should name the person who can accept that exposure, the temporary restriction, the evidence still required and the event that ends the exception. A date can help, but some exceptions should end immediately when scope, data or authority changes.

Do not use expiry to create an automatic approval. When an exception expires without the required evidence, the workflow should return to a defined restricted state or require a new explicit decision. If the system cannot enforce this, assign an operational control and verify that it actually occurs. A calendar reminder alone does not change production behavior.

Exceptions should also be easy to withdraw. Suppose a tool connector is allowed temporarily for a small internal group while its audit export is assessed. If the vendor changes its logging behavior, the original rationale may no longer hold. The owner needs a route to suspend the connector without renegotiating the entire AI program.

Track patterns across exceptions. Repeated requests for the same departure can mean that the policy is impractical, the delivery platform lacks a needed control or teams misunderstand the rule. Treat the pattern as a design question. Simply rejecting requests more firmly may not address the underlying cause; approving all of them quietly abandons the boundary.

A monthly governance meeting can therefore focus on decisions: which exceptions should close, which conditions need redesign, which workflows changed materially and which incidents exposed a weak assumption? Attendance and document counts are activity measures. They do not establish that the organization can control agent behavior or resolve disputed decisions.

Worked example: an agent that closes customer cases

Illustrative scenario. A service organization introduces an agent that classifies incoming messages and drafts responses for advisers. It then proposes automatic case closure when the agent believes the issue is resolved. The scenario is constructed to demonstrate decision design; it does not describe a Softlander client or measured outcome.

The original charter permits classification and drafting, with advisers responsible for sending responses and closing cases. Automatic closure changes the customer's access to further attention and the operational measurement of resolved work. The change is therefore treated as a new decision right, even if it requires only a small code change.

The workflow owner first narrows the candidate scope. Cases involving complaints, unresolved identity checks or inconsistent account records remain human-controlled. For the remaining scope, the team identifies what "resolved" means in the authoritative case system. A polite response from the customer is not automatically sufficient; the underlying request may still be incomplete.

The evidence review separates three questions. Can the agent recognize the permitted case type? Does the required business state actually exist? Can an incorrectly closed case be reopened and routed without losing context? The team creates cases where a conversation sounds complete but a required action has failed. It also tests missing records, duplicate messages and a customer's explicit request for further help.

The release authority has several options. It can keep all closure manual, permit the agent to recommend closure, allow bounded closure with a visible reopen route, or postpone the feature. The decision should compare operational burden and consequences rather than assuming that more autonomy is always progress. In this example, the chosen design is a recommendation queue while the team gathers evidence about the disputed states.

A proposed temporary exception would allow closure when the backend status is unavailable but the conversation seems resolved. The owner rejects that exception because it removes the very check that defines resolution. Instead, unavailable status routes the case

to an adviser. This is an example of governance changing behavior rather than merely documenting a risk.

The team also revises the business metric. Counting closed cases alone could reward premature closure. It pairs closure volume with reopened cases, unresolved repeat contacts and sampled review of completion state. These are suggested measures for this scenario, not validated targets. The owner chooses thresholds only after examining the actual workflow and consequences.

Keep governance connected to engineering and operations

A governance decision should be traceable to implementation. If a release permits only read-only access, the deployed identity and tool set should reflect that restriction. If a human must approve an external message, the execution path should require that approval. Ask the delivery team to demonstrate the condition in the running system, rather than relying solely on policy statements.

NIST's Secure Software Development Framework recommends integrating security practices into the software lifecycle. [5] For agent systems, use the existing development pipeline to preserve approved configurations, test evidence and change history. Governance supplies the decision conditions; engineering makes those conditions observable and repeatable.

Operating signals should reopen decisions when relevant assumptions change. Examples include a new model version, altered tool authority, a different data source, a new customer population or a repeated failure pattern. Not every change needs a board meeting. A pre-agreed classification can route routine changes through normal controls and material changes to the accountable owner.

Incident response must include the ability to reduce or withdraw agent authority. NIST SP 800-61 Revision 3 integrates incident-response considerations into wider cybersecurity risk management. [6] In a practical agent runbook, define who can disable a connector, switch to advice-only mode, preserve evidence and reconcile uncertain actions. A model rollback may not undo messages already sent or records already changed.

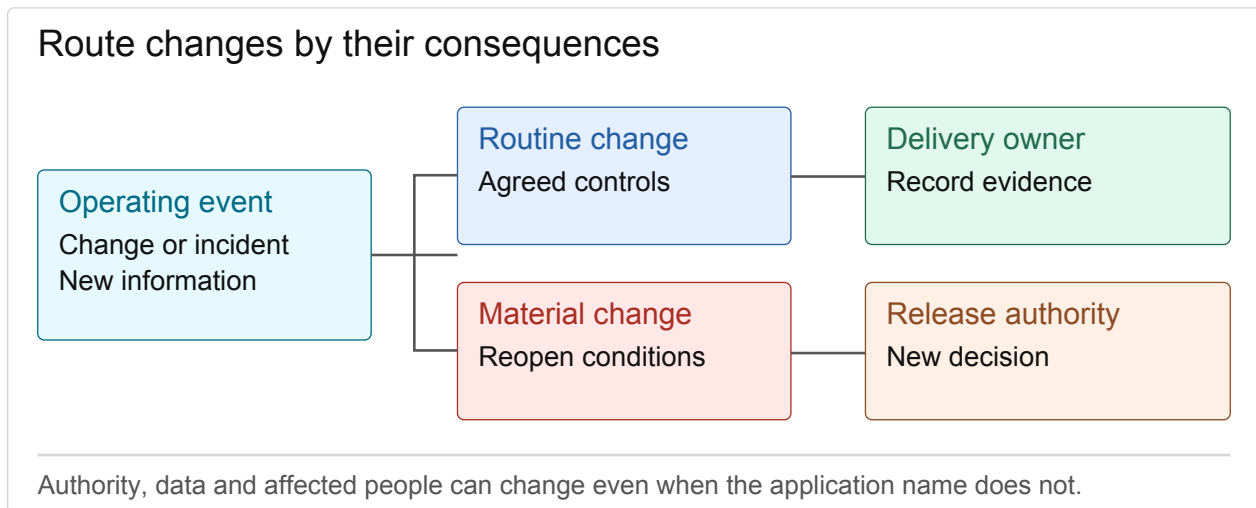


Figure 3. Routine and material changes follow different routes while preserving accountable decisions. Original Softlander conceptual diagram, 2026-09-16.

The governance record should survive team changes. Keep decision rationale, material objections and links to evidence in a durable location. A new owner should be able to understand why a restriction exists without reconstructing a conversation across chat messages. At the same time, avoid copying sensitive raw evidence into broadly accessible governance records; retain a sanitized decision summary and controlled links.

Test the process with a disputed decision

A governance model can look complete until a team disagrees with its conclusion. Exercise that situation before relying on it. Choose a realistic proposal that has business value but incomplete evidence, such as allowing a drafting agent to send external messages. Ask participants to state their recommendation, the evidence that would change it and the authority they believe they hold.

Observe whether the discussion produces a clear decision or simply another meeting. Can the owner constrain the scope while evidence is gathered? Can a specialist record a material objection without becoming the sole business decision-maker? Can an operator identify what must change in the running system? These are practical tests of the operating model itself.

Record any ambiguity as a process correction. The correction may be a clearer mandate, a missing escalation route or a better release artifact. Avoid rehearsals that assume agreement: the model is most valuable when commercial urgency and unresolved consequences point in different directions. A useful process preserves the disagreement and still identifies who must decide, on what basis and within which limits.

A working agenda for the next review

Choose one live or proposed agent workflow and bring its business owner, technical owner and operational owner together. Use an existing design or delivery review if possible. The purpose is to resolve a small number of decisions, not to launch a new program before the first workflow is understood.

Begin with the action boundary. List what the agent can recommend and what it can execute. Mark actions that affect money, access, personal information, external commitments or a person's ability to obtain help. Identify which of those actions already has an organizational approval rule. Where no rule exists, assign a decision owner instead of allowing the implementation to become the default policy.

Next, inspect the release conditions. For each important condition, ask for the corresponding implementation and evidence. If "human review" is a control, examine what the reviewer sees and what happens when they do not respond. If "restricted data" is a condition, inspect retrieval and logging paths. If "reversible" is a claim, demonstrate the reversal and identify effects that cannot be undone.

Then examine exceptions and changes. Which restrictions are temporary? What ends them? Which recent changes altered the original assumptions? Which operational signals would require the workflow to stop? Record the answer in language that an on-call operator can use, and confirm that the responsible person has the access and authority to act.

Close with a decision: allow the defined scope, constrain it, hold it for specific evidence, or stop it. Give each open action an owner and completion condition. Do not convert uncertainty into approval by averaging scores or assuming that a lack of incidents proves adequacy. Equally, do not require perfect knowledge before permitting a genuinely bounded, recoverable use.

The most useful governance system is one that people can use at the moment of change. It makes authority visible, preserves reasons, respects specialist judgment and provides a practical route to correct course. Start with one workflow and demonstrate that the process can alter a release decision. Expand the operating model when that usefulness is evident.

References

- [1] NIST. Artificial Intelligence Risk Management Framework 1.0. January 2023.
- [2] OECD. AI Principles. Adopted 2019; updated May 2024.
- [3] NIST. SP 800-53 Revision 5: Security and Privacy Controls. September 2020, with release updates listed by NIST.
- [4] NIST. AI 600-1: Generative Artificial Intelligence Profile. July 2024.
- [5] NIST. SP 800-218: Secure Software Development Framework 1.1. February 2022.
- [6] NIST. SP 800-61 Revision 3: Incident Response Recommendations. April 2025.