

Own the Learning Loop

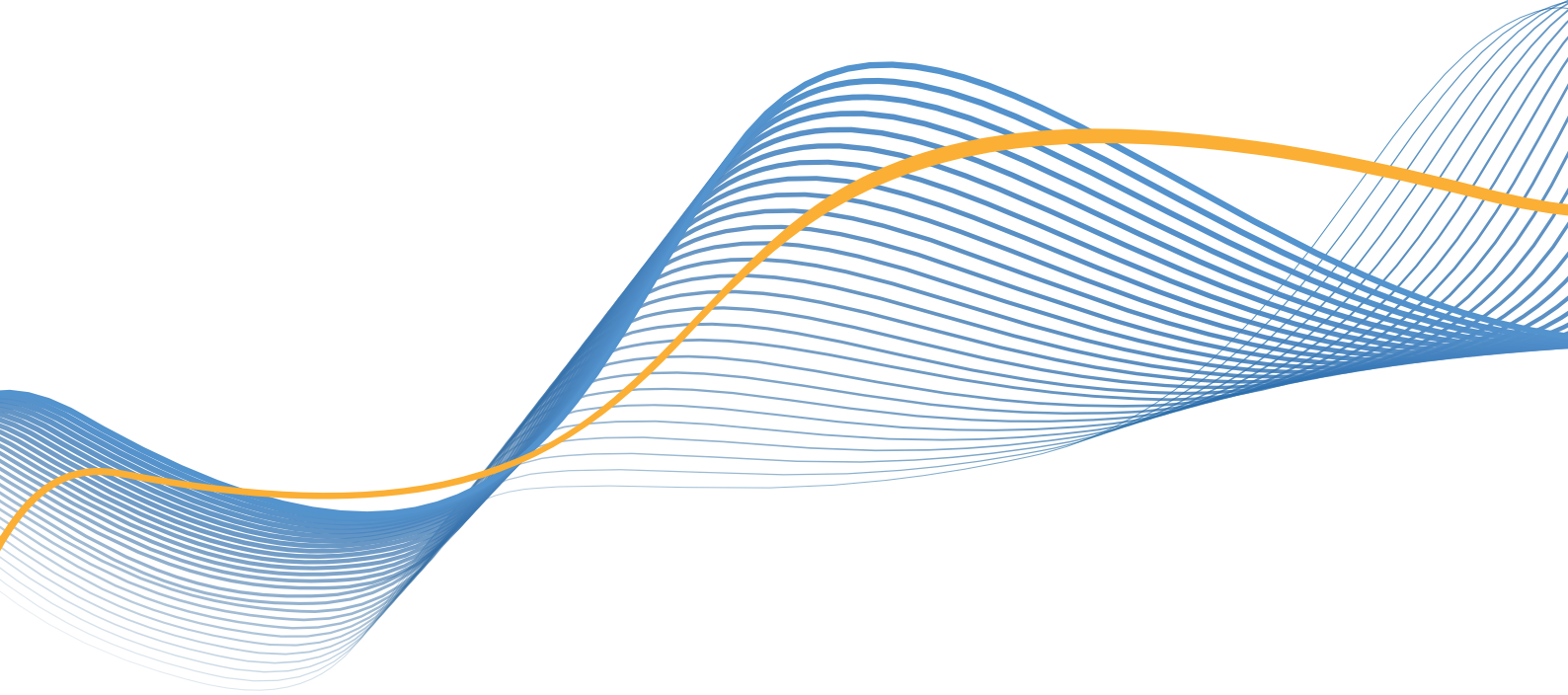
Composing the AI-First Insurer:
what to own, rent, or share



Third in the AI-First Insurance series. “An AI-First Insurance Blueprint: From pilot sprawl to compounding advantage” set out why the AI-first operating model is materially different from AI-enabled, and what architecture it requires. “Archetypes: Starting Points for the AI-First Insurance Journey” examined a carrier’s starting point. The Blueprint called the effect of connected learning loops a flywheel. This piece asks what has to be owned for the learning loops underneath it to close. The three pieces can be read in any order.

Table of Contents

- **Why ownership is the question**
- **Part I – What defines a loop, and where does it break**
 - Figure 1. The learning loop and where it commonly breaks
- **Part II – What has to be owned: three modes of control**
- **Part III – Systems of Record, Intelligence, and Orchestration, and the Integration Layer**
- **The question is no longer build versus buy**
- **Appendix**
 - Exhibit A — Do You Have a Loop? A self-assessment
 - Exhibit B — Own / Rent / Share Decision Card
 - Exhibit C — The Systems Architecture Map



Why ownership is the question

In an AI-first insurer, each decision makes the next one better. An underwriting judgment refines next quarter's appetite. Claim outcomes correct a pricing assumption. A decline whose reasoning is captured improves the next triage decision. These are enterprise learning loops, and they are the essence of AI-first insurance.

The carriers that pull ahead will not be those with the largest model inventory or the buzziest AI-assisted platform. They will be those that can close one learning loop cheaply enough to close the next related one. That ability depends on something that should not be rented or shared: ownership of the institutional intelligence the loop uses and produces. Not the tools or services: the accumulated insight is what differentiates one carrier from another.

Ownership means retaining the rights, records, semantics, and decision authority required to keep the learning loop functioning when a vendor or platform changes. The carrier must be able to move not just the decision itself, but also the reasoning behind it, the rules in force, the approval history, and the link to the eventual outcome. Ownership does not mean building or operating every component. It means ensuring that the institutional intelligence created through those components remains under the carrier's control and can continue to be used when the surrounding technology changes.

In other industries a loop can close on its own: observe the outcome, retrain, redeploy. For insurance, and for financial services generally, this operating model is meaningfully harder, because a human gate sits inside every loop that touches a filed or regulated decision. Actuarial sign-off. Model validation. Adverse action explainability. That gate is a design constraint, not friction to be engineered around or out. The key question for executive leadership is whether sign-off entails a review or a three-month forensic reconstruction.

But a trap awaits: the first loop can always be forced through by hand. Assemble the evidence, rebuild the basis, walk it through validation. It works, it gets the model into production, and the organization learns what closing a loop costs. The trap is allowing the second loop—and every loop after it—to require the same reconstruction. If the first loop leaves behind no reusable decision record, outcome link, or approval trace, you have a bespoke artifact rather than a capability. The effort does not carry forward, so each subsequent loop is as costly as the first—and eventually harder.

The reason is structural. A decision is made inside one vendor's system and the outcome lands in another's. Neither may record the reasoning behind the decision, and without a shared key the carrier cannot reliably connect them. **You cannot reliably close a loop across a boundary whose records, rights, and interfaces you do not control.** Ownership is therefore an operating-model decision, not merely a procurement task: the carrier must control the records, rights, and connection points that let it learn repeatedly without reconstructing the basis each time.

Composable architecture preserves the carrier's ability to join decisions, outcomes, reasoning, and approvals as individual systems change. The carrier owns the linkages between decisions and outcomes, the semantics, and the decision records; surrounding components remain replaceable. But composability alone does not determine which components should be durable.

Today, the right question is: own, rent, or share?

This piece is about the architecture of an AI-first insurer: which capabilities it owns and how its systems must be composed for a decision to keep improving instead of merely repeating. Part I defines a learning loop and identifies where it typically breaks. Part II gives a framework for deciding which capabilities to own, rent, or share. Part III applies that framework to the systems an insurer runs, and the governance question that follows. It does not address whether the people running these loops are ready to operate them, which matters at least as much.

The three exhibits in the appendix move from understanding to first steps:

- the self-assessment scores an existing initiative against the five steps and shows which gap to close first;
- the decision card works one component through the own, rent, or share test;
- the architecture map shows the mode of control that applies to each component across the systems.

No carrier has built this end to end. What follows is a point of view on how to get there, not a proven playbook.

The prescription, in most cases

OWN

what improves through your own experience: product definitions and rating logic; lifecycle rules such as mid-term adjustment, renewal, and cancellation conventions; appetite and risk selection; models trained on your own outcomes, meaning the weights and the training data rather than access to them; the routing logic that decides what goes straight through and what escalates; and the decision records that make any of it joinable later.

RENT

what improves without your experience: the policy administration platform and its transaction engine; the regulatory content subscription covering bureau content, forms, state filings, and statutory formats; foundation models and inference infrastructure; commodity extraction and classification; agent frameworks and protocol implementations, where the framework executes while the routing rules and authority limits stay owned.

SHARE

what improves through the collective experience of the market: fraud consortium data, catastrophe models, industry loss and mortality experience. Pooling only works where exclusivity makes the asset weaker, and contribution still requires explicit terms on permitted use, confidentiality, and withdrawal.

PART I – What defines a loop, and where does it break

The learning loop is the mechanism by which AI-first insurers improve and adapt continuously. At minimum, each loop has five steps:

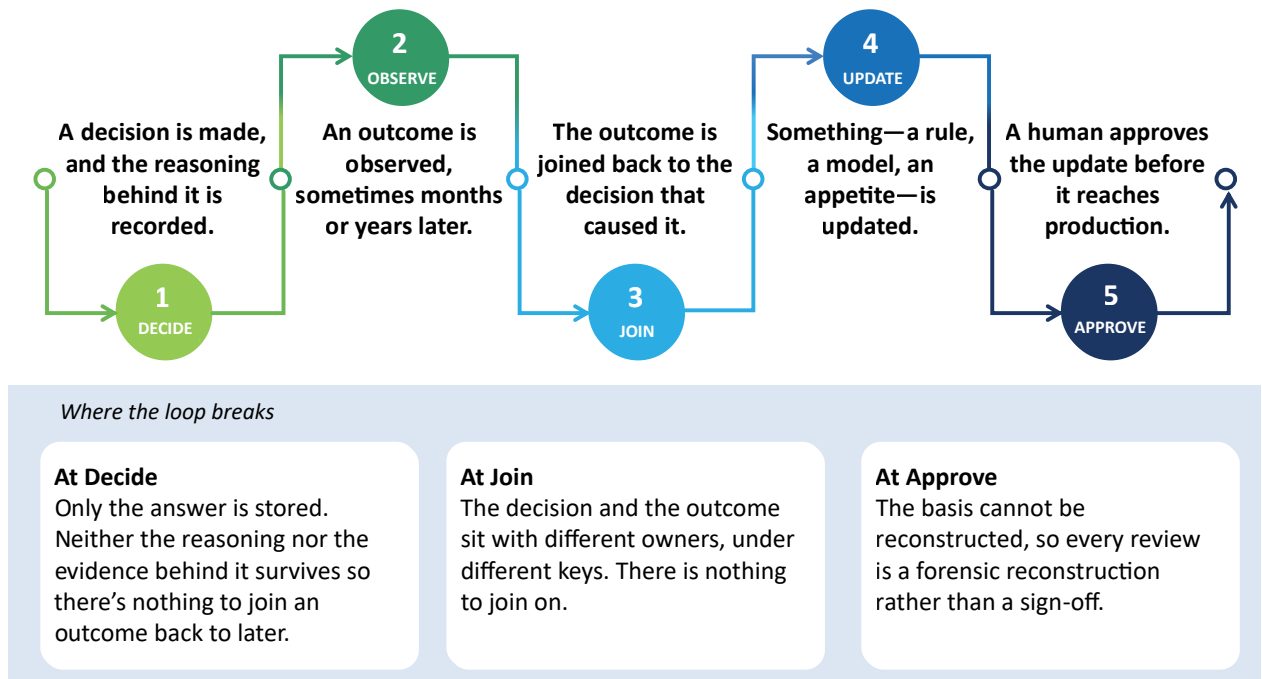


Figure 1. The learning loop and where it commonly breaks

Individual decisions may require additional controls for monitoring, challenge, deployment, exception handling, and rollback.

Each loop type runs across many individual decisions, each of which is an instance of the loop.

Referral outcomes feeding triage, or decline reasons feeding appetite, are distinct loops. Every submission that meets a loop's criteria creates an instance of it. A large multi-line insurer may observe thousands of loop instances a day, but the number of loop types remains countable. Confusing the type with its instances makes the inventory feel impossible. It is not: it is a finite set of patterns, run consistently.

A workbench flags a referral and records its trigger. Weeks later, the referral is bound, modified, or declined. If a trigger routinely produces acceptable outcomes, the carrier can deprioritize it after approval. The outcome has improved the next triage decision.

The five steps look simple, but breakdowns are common, usually at the same three steps.

Decide.

Systems often record the decision but not its reasoning. Without that context, a regulated enterprise may see how a trigger performs but cannot determine responsibly why it should change.

Join.

The referral sits in one system and its resolution lands in another under a different identifier. What appears to be a data problem is often a contracting failure.

Approve.

A proposed change cannot be judged without reconstructing what the system originally saw. That reconstruction—not the approval itself—is the avoidable cost of the human gate.

5

Left unfixed, a break affects not just one decision but every instance of that loop type in every division that runs it. This is the hidden reason many AI initiatives stall at the pilot stage: it is not that the technology cannot scale, but that a broken loop type prevents it from doing so.

Worse than having no key is having one that mostly works. A carrier joins referrals to outcomes on the submission number, so the reporting shows which triggers produce acceptable outcomes. The join holds until an endorsement, a resubmission through a second broker, or a renewal gives the outcome a new number. Those are not edge cases in specialty, and rewritten ones are often the complicated cases, which are the referrals the loop most needs to learn from. The reporting still looks healthy because it only counts the referrals that joined, and those are the simpler ones.

One open question: What is the right number of loop types? No carrier has made the full transition, so nobody knows. What is likely is that carriers start with a small number of high-consequence decisions because every loop requires scarce approval capacity: actuarial sign-off and model validation. The frontier then expands into lower-impact decisions as the cost of closing a loop falls.

Two points follow. First, the easiest loops to close are usually contained within one function, where outcomes are easy to see. But those are not always the loops with the most valuable learning. If carriers choose loops function by function, they will favor quick feedback over important outcomes. Second, the most valuable loops often cross several functions and take longer to close. For example, what a carrier declines—and how quickly—affects what brokers submit next. That changes the portfolio mix, which influences the terms reinsurers offer at renewal. This loop spans distribution, underwriting, and ceded reinsurance, and may take years to complete.

Exhibit A scores an existing initiative against the five steps and shows which gap to close first.

You cannot reliably close a loop across a boundary whose records, rights, and interfaces you do not control.

PART II – What has to be owned: three modes of control

Build versus buy used to be a tolerable proxy for own versus rent. Build usually meant owning the artifact; buy usually meant renting access to someone else's. The proxy worked because the high cost of building naturally rationed in-house development to high-value technology. Software ran on the carrier's infrastructure, so the records remained within reach regardless of the license.

AI breaks the first assumption by making it cheap to build. SaaS breaks the second by moving software and infrastructure outside the enterprise, along with control of the underlying systems and data. The relevant distinction is not how a component is acquired, but its mode of control:

- Own.** The carrier must hold durable rights to modify and retain the component and be able to move it—with its context and history—into another usable environment. The decision records behind the referral trigger are the clean case: captured inside a vendor's workbench, they are owned only if they can leave it and be read somewhere else. Ownership is not authorship: own whatever the loop depends on, regardless of
- Rent.** You pay to use a capability without acquiring the underlying intellectual property. This is the right mode wherever improvement depends mainly on someone else's investment rather than on the carrier's own experience. The foundation model used by the workbench to read the same submission is the clean case: it improves largely through the vendor's roadmap and is broadly available to its customers. Very few carriers should own one.
- Share.** Contribute to and draw from a pooled asset where collective experience makes it stronger, as with fraud consortium data screened against the same submission, catastrophe models, or industry experience. Sharing still requires explicit rules for permitted use, confidentiality, provenance, withdrawal, and how pooled insight enters the carrier's own loops.

Renting the loop is defensible for a while. Renting it indefinitely is not, since every competitor on that contract improves at the same rate.

Own, Rent, and Share cover the components carriers are deciding about today. For anything else, ask (1) whether it can move, (2) whether a loop has to close over it, and only then (3) whether to build it. Answering the third question first is how carriers end up owning things they cannot move. A carrier that builds its own rating logic inside a vendor platform's scripting layer has built it and would say it owns it. But it is written in that vendor's rules language against that vendor's data model, so at migration it cannot be lifted, only rewritten. Building settled whether the carrier could have it. It never settled whether the carrier could keep it.

Adjacent loops improve different decisions while reusing records, identifiers, or evidence that an earlier loop already established. The referral-triage loop is adjacent to the decline reasoning-appetite loop. They answer different questions, but both rely on the same submission, decision, and outcome records. That distinction matters because it reveals whether the carrier has built a capability or merely completed a project. If the second loop can be implemented at materially lower cost than the first, reusable foundations exist. If it costs roughly the same, the first loop left little behind. In practical terms, the second loop should be able to read existing decision records and add only the information needed for its own purpose, rather than reconstructing the evidence from scratch.

Exhibit B works one component through the own, rent, or share test, with the portability checks from this section.

PART III – Systems of Record, Intelligence, and Orchestration—and the Integration Layer

To determine the appropriate mode of control, apply the framework from Part II through a one-question test: **What causes this component to improve?**

- Own if it improves through the carrier's own institutional intelligence and experience.
- Rent if it improves through a vendor's product roadmap and feature releases.
- Share if it improves through the collective experience of the market.

Systems of Record (SoR). Rent the transaction engine. Rent standardized bureau and statutory content, but own proprietary product logic, appetite rules, and decision history—even when the platform that authors or executes them is rented. Institutional intelligence must remain accessible independently of the system that recorded it.

Where that split falls depends on the book. Most admitted product content is standardized, leaving a small set of owned deviations and appetite rules layered on top. In E&S, freedom of rate and form means most of it is proprietary by construction. The distinction is visible in the market already, as product configuration is increasingly externalized from the policy administration systems (PAS) in separate tooling.

The boundary is moving. Specialty content was manuscripted because no library existed to rent. As platform vendors build specialty form and rate libraries, that may no longer be true and content a carrier previously owned out of necessity could then be rented. The judgment did not change; the market did. A carrier consolidating platforms should re-ask what causes this component to improve rather than lifting and shifting a manuscript library it may no longer have to maintain.

Systems of Intelligence (Sol). Rent the foundation model, but own the carrier-specific intelligence built around it: tuning data, evaluation evidence, decision policies, and approval history. This matters more in specialty than elsewhere. Thin volumes limit what a generic model can learn, so the accumulated evidence about how this carrier's decisions performed may be the only thing a competitor cannot buy on the same terms.

For example, a rented model may extract facts from a submission, but the carrier must retain what it extracted, its confidence, what the underwriter modified, why, and how the account later performed. Over time, those records become a source of institutional intelligence. They reveal which errors matter, when human review is warranted, and how decision-making should evolve. In specialty lines, where volumes are often thin, that accumulated evidence may be more valuable than the model itself.

A carrier that tunes a vendor's model on its own outcomes has produced something valuable and may hold no rights to it. If the tuning data and evaluation evidence live only in the vendor's environment, switching models means starting the evidence over, which is what makes the switch expensive rather than the model itself.

Systems of Orchestration (SoO). Rent the plumbing but own the judgment. API orchestration improves through vendor enhancement; routing rules, confidence thresholds, escalation logic, and final authority improve through carrier experience. If that agentic logic exists only as vendor configuration, it may disappear during migration.

A confidence threshold that sends marine cargo referrals to a senior underwriter but clears property schedules straight through may look like a setting, but it represents years of underwriting judgment.

Integration Layer. System integration is not enough. The carrier must be able to connect a decision to its eventual outcome even when that outcome arrives years later or the platforms in between have changed. That requires three things.

1. Carriers solve the linkage problem in different ways: a decision key that follows the transaction downstream or agents that trace lineage. The outcome matters more than the approach: a reviewer must be able to track a decision to its result without rebuilding the basis for the decision. Otherwise, the cost of the human approval gate rises sharply because every review becomes an investigation rather than a sign-off.
2. Recording rules, which belong in the vendor contract: what must be written on every decision, in what form, and that the carrier can extract it in full, including the history of changes and not only the current state.
3. Retention that covers the basis. Statutory schedules preserve the transaction. The loop needs the reasoning, the rule or model version in force at the time, the approval trail, and whatever carries the linkage. In long-tail lines that has to outlast the platforms and often the vendor relationships.

The carrier need not own every system, but it must own the linkage between them.

Following the referral loop shows how mixed a single loop is. The underwriting workbench is a product, not a system. The learning loop running through it touches all three. Inside the System of Record, the transaction is rented while the eligibility and referral logic is owned, because that logic encodes which submissions this carrier's underwriters judged worth their time. Inside the System of Intelligence, the model that reads the submission is rented while the record of what it extracted and what the underwriter did with it is owned. Inside the System of Orchestration, the API plumbing and the agent framework are both rented, while what the agent is permitted to decide, at what confidence, and where it must stop are owned.

Weeks later the referral resolves in the policy system, and nothing in those three closes the gap between the resolution and the trigger that caused it. The Integration Layer closes it, or the loop does not close. One loop, three systems, and the mode of control differs by component inside each of them.

Trust is embedded, not another system. Identity and authority establish who or what may act; guardrails constrain that action; evidence and provenance record its basis; and accountability records approval, override, and outcome. These controls must be captured as the decision is made and should be proportional to its consequence.

Trust also changes what must be governed. Model governance remains necessary, but it is no longer sufficient. It asks whether an individual model remains valid. Loop governance extends that

discipline to the full decision system. Is the decision still producing the intended outcomes? Are its evidence and authority intact? Are approved changes reaching production? Is the loop continuing to learn?

A model can remain technically valid while the decision it supports stops improving. Its controls may still pass because those controls were designed to test the model, not the wider system in which it operates.

Model governance watches the model. Loop governance watches whether the full decision system is still learning.

Exhibit C maps the components of each system and the integration layer against the own, rent, or share test, with the trust controls that have to be embedded across them.

The question is no longer build versus buy

The industry has focused heavily on which AI capabilities to buy. That is no longer the decisive question. Capability is becoming abundant and cheap, and a capability a competitor can acquire on the same terms is not a moat. What remains scarce is the ability to improve repeatedly in a regulated setting, at a cost the enterprise can sustain.

That ability is not a technology. It is a property of how the operating architecture is assembled: whether decisions carry their reasoning, whether outcomes can be linked back to them, whether the human gate is a review or a forensic reconstruction, and whether the carrier owns the record that connects decisions, outcomes, reasoning, and approvals, or whether that connective record belongs to someone else.

The AI-first insurer will not be the one with the most AI. It will be the one whose AI makes the next decision cheaper to improve and easier to approve. Start with one consequential, repeated decision: map its loop, identify the missing record, right, or control point, and set the ownership requirements for the systems it crosses. The question is not what to automate. It is what to own so the enterprise can keep learning under its own power, at its own pace.

Authors

Howard Kahai Pang

Consulting Partner – Insurance, TCS

Vince Ofrecio

Managing Partner – Insurance, TCS

Lino Vivas

Consulting Partner – Insurance, TCS

EXHIBIT A: Do You Have a Loop?

A self-assessment for an existing AI initiative

Most AI pilots in insurance build the first step well and stop there. This tells you how far a current initiative is from a closed learning loop, and which gap to close first.

Score each step against the initiative you have today, not what could be built if someone spent a quarter on it.

Accessible

Available now, without special effort.

Extractable

Exists somewhere, but getting to it takes real work.

Inaccessible

Cannot be reached, whether or not it exists.

The five steps

1. **DECIDE – Is the reasoning behind the decision accessible?**

Not that a decision was made, but why. The trigger that fired, the underwriter's stated basis, the evidence in front of them at the time.

☐

Accessible

☐

Extractable

☐

Inaccessible

2. **OBSERVE – Is what happened afterward accessible?**

Whether the account bound, how it performed, whether the claim developed. Without chasing it down.

☐

Accessible

☐

Extractable

☐

Inaccessible

3. **JOIN – Is the connection between the decision and its outcome accessible?**

A shared identifier that survives the gap between them, rather than someone matching records by hand.

☐

Accessible

☐

Extractable

☐

Inaccessible

4. **UPDATE – Is the path from a finding to an actual change accessible?**

Whether a pattern can reach the rule, model, or appetite it should change, without a special project to carry it there.

☐

Accessible

☐

Extractable

☐

Inaccessible

5. **APPROVE – Is the evidence for sign-off accessible?**

Whether a reviewer can see what the model saw, or has to reconstruct it before judging whether a change is sound.

☐

Accessible

☐

Extractable

☐

Inaccessible

Reading the result

Mostly Accessible is rare and means a working loop. A mix of Accessible and Extractable is the common result: the pieces exist, but nothing connects without someone doing it by hand. Mostly Inaccessible means what you have is decision automation, not yet a loop.

The number of Accessible rows is not the finding. The finding is which single Extractable or Inaccessible step is keeping the next loop from getting cheaper. That step is where the work is.

Before you pick the next loop

Before selecting the next loop, test for adjacency. An adjacent loop improves a different decision while reusing records and identifiers established by the first.

- **A new field** on a record you already write is cheap. You are capturing one more thing at a decision point you already instrument.
- **A new join** is not. It means reaching a system that is not yet connected, with an identifier that has to survive the interval between decision and outcome.

Ask three questions of any candidate:

1. Does it improve a genuinely different decision, or the same one relabeled?
2. Does it read records the first loop already captures, adding at most a field or two?
3. Does it use the existing identifier, or does it need a new join?

A different decision, mostly existing records, and the same identifier mean the loop is adjacent. Plan it as an extension. A new join means it is a first loop of its own and should be planned and priced accordingly.

The broader implication: Enterprise coverage develops through a small number of increasingly rich domain records that are eventually connected. Loops can expand through adjacency within a domain; connecting domains usually requires a new join and should be treated as separate work.

This shows where an initiative stands. Building the missing steps for a specific system is a different exercise, and the one we run with clients. Use with Exhibit B to decide what owning the fix requires.



EXHIBIT B: The Own / Rent / Share Decision Card

Take one component. Work through it in order. Stop when you reach a verdict.

Component under review:

Which loop or loops depend on it:

Step 1 — The improvement test: What causes this component to get better?

If it improves through...	The mode is...
Our own institutional intelligence — our decisions, our loss experience, our underwriters' judgment	OWN
A vendor's product roadmap and feature releases, identically for every customer holding the same contract	RENT
The collective experience of the market, and it gets stronger the more participants contribute	SHARE

If two answers seem right, the component is not one component. Split it and run each half through separately. Product logic splits this way: standardized content improves when the bureau publishes and the vendor implements; manuscript wordings, eligibility rules, and rating structures improve because of our own experience.

Answer (Own / Rent / Share / Splits — list the parts)

Step 2 — If the answer is OWN, test whether we own it in practice.

Ownership is a claim about portability, not about who built it. Both tests have to pass.

Legal

- Do the rights convey to us under the current contract, or do we hold access only?
- Do they survive the vendor being acquired, sold, or wound down?
- Is derived material — models tuned on our outcomes, records generated in the vendor's system — covered, or only material we supplied?

Technical

- Can the artifact leave in a form something else could read?
- Does it carry its context and history with it, or only its current state?
- If we exported it today, what would be missing that a loop needs?

Verdict (Owned in practice / Nominally owned, not portable / Not owned):

.....

What has to change at the next renewal:

.....

Step 3 — If the answer is RENT, name what stays ours anyway

Renting the capability is correct. Renting what our own experience produces through it is not.

- What records does this component generate about our decisions and their reasoning?
- Where do those records live, and can they leave?
- What has this component learned from our data, and do we hold it or only benefit from it?
- What is the vendor liable for — currency, correctness, filing accuracy — and is that in the contract?

What we own alongside the rental:

Step 4 — If the answer is SHARE, define the boundary

- What are we contributing, and under what terms?
- Does contributing expose anything about our appetite or pricing that we would not disclose otherwise?
- What do we get back that we could not build from our own book alone?

Trap to check for: Does this component look like configuration while containing accumulated judgment?

Agentic orchestration is the common case: confidence thresholds, escalation rules, straight-through eligibility, and authority limits read as settings in an admin screen and represent years of underwriting decisions. So do appetite rules and referral triggers.

Yes / No — if yes, treat as owned and confirm it can leave the platform:

Verdict

Mode	Own / Rent / Share
Owned in practice today?	Yes / Nominally / No
What has to change	
When	Next renewal / Migration / Now

Worked example

One component, carried through the whole card. The answer that matters here is the one in Step 2 — the mode was never in doubt; the portability was.

Component under review:

Agentic orchestration in the submission triage workbench — straight-through eligibility, confidence thresholds for escalation, referral routing, authority limits.

Which loop or loops depend on it: Referral outcomes to triage. Declination reasons to appetite reads the same records.

Step 1 — The improvement test

OWN. The thresholds encode which submissions our underwriters judged worth their time and which they did not. The vendor’s roadmap does not move them; our own referral outcomes do.

Step 2 — Portability

Legal: partial. The MSA conveys rights to data we supplied. It is silent on routing logic authored inside the vendor’s configuration UI and on records the platform generated about our decisions. Nothing addresses what survives if the vendor is acquired.

Technical: fails. Current thresholds export to a settings file. The history does not — no record of what each threshold was before, why it changed, or which referral outcomes prompted the change. Export gives us the current state and none of the reasoning, which is the part the loop runs on.

Verdict: nominally owned, not portable.

Step 3 — What must remain portable alongside the rented platform

The platform is rented and should stay rented; the transaction engine is not the issue. What has to come with us is the decision record behind every routing action: which trigger fired, on what basis, and how the referral resolved.

The trap

Yes. This is the case the trap describes. Four years of underwriting judgment sit in an admin screen that looks like configuration, and a migration would carry the settings over as numbers with no record of why any of them are what they are.

Verdict

Mode	Own
Owned in practice today?	Nominally — rights are partial, history is not exportable
What has to change	Export path for threshold change history with the outcomes that prompted each change; contract language covering logic authored in the vendor’s UI and records generated about our decisions; survival on change of control
When	Next renewal — before any migration is scoped, not during it

EXHIBIT C: The Systems Architecture Map

Shows how the modes of control typically apply across system components. Compare the map with your own component inventory.

How to read this map

The components listed are illustrative, not exhaustive, and the modes show where the improvement test usually lands—not where it must land. A specialty carrier that concludes it should own something shown here as rented has probably applied the test correctly to a component that behaves differently in its market. Exhibit B resolves anything not named here, and anything named here that does not fit.

Systems of Record

Component	Mode	Why
Transaction engine	Rent	Improves on the vendor's release schedule, not on our underwriters' decisions
Standardized product content — bureau forms, statutory rates, filing formats	Rent	Improves when the bureau publishes and the vendor implements; vendor liability for currency and correctness is part of what is bought
Proprietary product content — manuscript wordings, eligibility rules encoding appetite, rating structures reflecting our loss experience	Own	Improves because of our own experience; own it even where the authoring and execution environment is rented
Decision history — what was declined and why, which terms changed, how referrals resolved, which decisions produced good or bad outcomes	Own	Institutional intelligence; must remain readable independently of whatever platform recorded it
Billing and receivables, reinsurance cessions, document generation, statutory reporting	Resolve individually	A policy administration platform contains additional components; apply the same test to each

Important qualifications.

The boundary varies by line of business and may shift as vendors expand their offerings. In long-tail lines, the reasoning and evidence behind a decision must also survive the platform that recorded it.

Systems of Intelligence

Component	Mode	Why
Foundation models and inference infrastructure	Rent	Improve on the vendor's roadmap, identically for every customer
Commodity extraction and classification	Rent	Same
Models tuned on our own outcomes — weights and training data, not access to them	Own	The value comes from our experience, not the architecture
Tuning data, evaluation evidence, decision policies, approval history	Own	This is the intelligence that makes a rented model specific to us
Fraud consortium data, catastrophe models, industry loss and mortality experience	Share	Stronger pooled than held alone; a model trained only on our book sees only our fraud

Systems of Orchestration

Component	Mode	Why
API orchestration — calling a rating engine, writing to a policy record	Rent	Improves through vendor enhancements
Agent frameworks and protocol implementations	Rent	Same
Agentic orchestration — what processes straight through, what requires review, what confidence threshold triggers escalation, who holds final authority	Own	Improves as we learn; looks like ordinary configuration while containing years of accumulated underwriting judgment

The Integration Layer

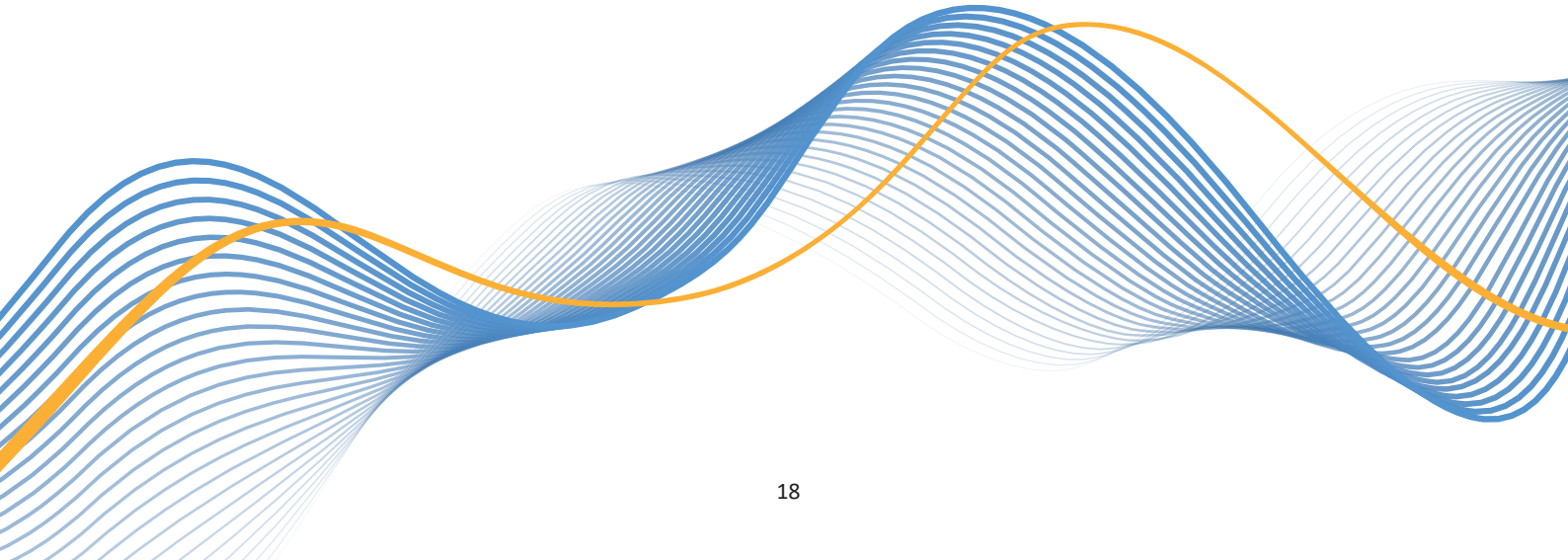
Component	Mode	Why
A durable way to identify a decision across platforms and over time	Own	Without it there is no key to join a decision to the outcome that tests it
Rules for what each system must record, and in what form	Own	The most common break in a loop is contractual before it is technical
Retention standards that preserve the linkage through a vendor or platform change	Own	The linkage has to outlive the systems that produced it

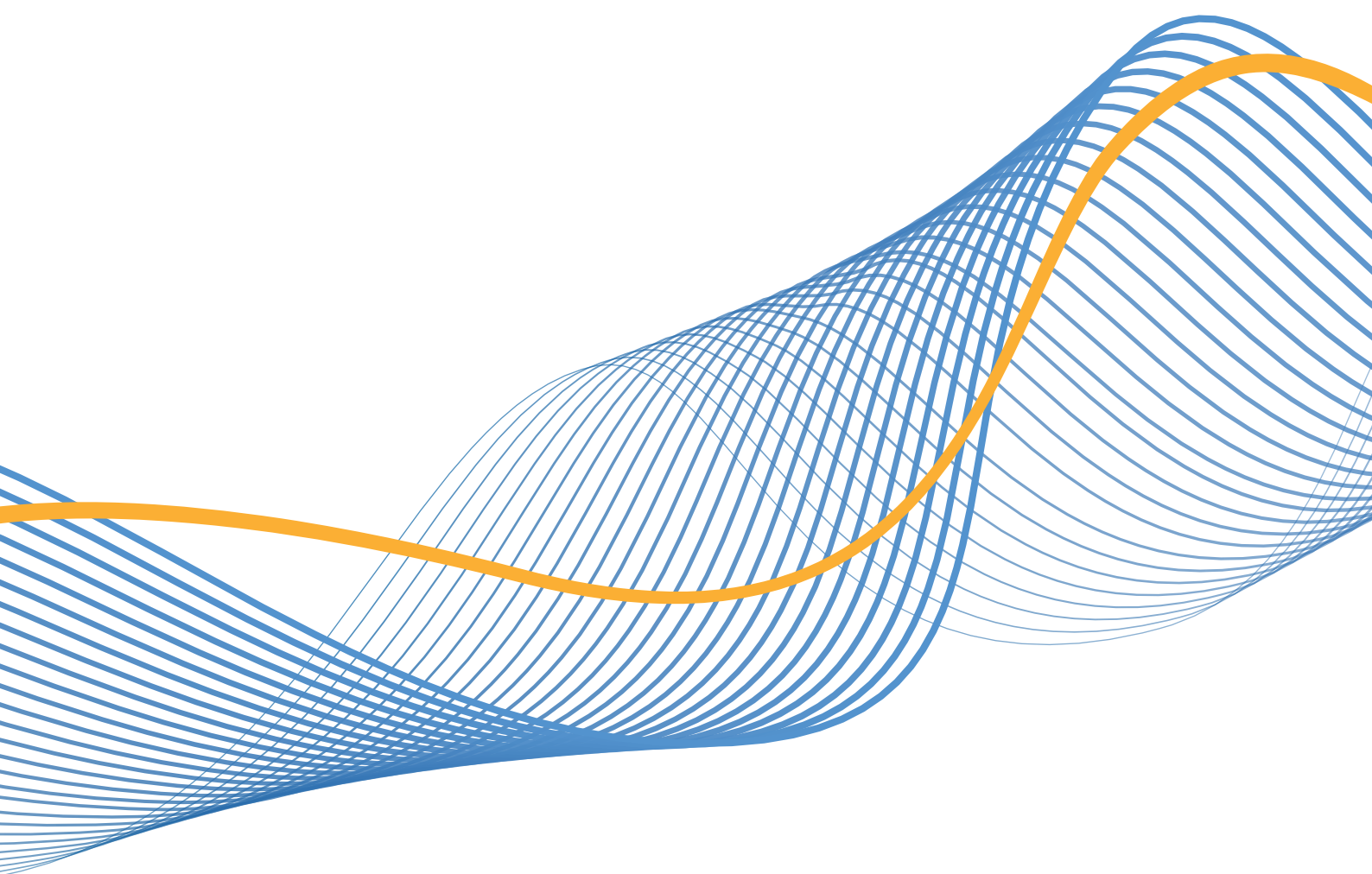
System ownership does not substitute for ownership of the linkage.

Trust — embedded across all four, not a fifth system

Control	What it establishes
Identity and authority	Who or what is acting, what it is permitted to do, and when human approval is required
Guardrails	Business rules, thresholds, and escalation requirements that constrain the action
Evidence and provenance	The data, model, instructions, and policies behind the decision
Accountability	Whether a human or an AI system decided, who approved or overrode it, and what happened afterward

Controls should be proportional to the consequence of the decision. An assistant drafting a communication needs different controls from an agent that can bind coverage, deny a claim, or change a reserve.





Contact

Visit <https://www.tcs.com>

About Tata Consultancy Services Ltd (TCS)

Tata Consultancy Services (TCS) (BSE: 532540, NSE: TCS) is a digital transformation and technology partner of choice for industry-leading organizations worldwide. Since its inception in 1968, TCS has upheld the highest standards of innovation, engineering excellence and customer service.

Rooted in the heritage of the Tata Group, TCS is focused on creating long term value for its clients, its investors, its employees, and the community at large. With a highly skilled workforce spread across 55 countries and 202 service delivery centers across the world, the company has been recognized as a top employer in six continents. With the ability to rapidly apply and scale new technologies, the company has built long term partnerships with its clients – helping them emerge as perpetually adaptive enterprises. Many of these relationships have endured into decades and navigated every technology cycle, from mainframes in the 1970s to Artificial Intelligence today.

TCS sponsors 14 of the world's most prestigious marathons and endurance events, including the TCS New York City Marathon, TCS London Marathon and TCS Sydney Marathon with a focus on promoting health, sustainability, and community empowerment.

TCS generated consolidated revenues of over US\$30 billion in the fiscal year ended March 31, 2026. For more information, [visit www.tcs.com](https://www.tcs.com)

Follow TCS on [LinkedIn](#) | [Instagram](#) | [YouTube](#).

All content/information present here is the exclusive property of Tata Consultancy Services Limited (TCS). The content/information contained here is correct at the time of publishing. No material from here may be copied, modified, reproduced, republished, uploaded, transmitted, posted or distributed in any form without prior written permission from TCS. Unauthorized use of the content/information appearing here may violate copyright, trademark and other applicable laws, and could result in criminal or civil penalties.

Copyright © 2026 Tata Consultancy Services Limited