INTELLIGENCE WAREHOUSE · RESEARCH

Decision circuits

A company's most valuable operating asset is the way it makes its recurring decisions: the thresholds, the exceptions, the sense of when a case needs to go upstairs. Almost none of that has a system of record. Some of it is written down, most of it is in people's heads, and the rest only shows up when the decision actually runs. This paper describes the decision circuit, the unit the Intelligence Warehouse uses to capture one recurring decision in executable form, and follows it through the four layers that fill it: documents, expert interviews, decision traces, and reinforcement from live usage. A seeded simulation measures what each layer contributes, and what it costs to skip one.

Authors Akhil Singh · Nidhi Prabhu · Aniruddha Tammewar **Date** September 2026

Method Seeded simulation, 420 circuits, 12 months **Reproducible** seed 20260919

81%	33.5%	1.9%	98%
OF INTERVENTIONS TRACE TO RULES NO DOCUMENT CONTAINS	INTERVENTION RATE IF YOU STOP AT DOCUMENTS	INTERVENTION RATE WITH ALL FOUR LAYERS, MONTH 12	OF THE TRAFFIC-WEIGHTED RULEBOOK CAPTURED BY MONTH 12

An intervention is a decision the system got far enough wrong that a person had to override, rework, or rescue it. About **38.3%** of the constraints that govern a company's decisions never appear in any document, and because those unwritten constraints are exactly the ones that decide the hard cases, they cause most of the failures. That asymmetry is the reason circuits are hydrated in layers rather than ingested once.

01 Where decision logic lives today

Ask where your company's data is and someone points at a warehouse. Ask where the logic that runs the company is, the actual rules by which discounts get approved and stock gets reallocated and exceptions get granted, and there is no equivalent answer.

The honest answer is that it lives in three places, none of them a system.

Some of it is written down. SOPs, policy PDFs, process wikis. This is the part everyone already agrees on, which is precisely why it made it into a document. It is also chronically stale: the process changed in March and the PDF did not.

Most of the rest is in people's heads. The pricing analyst who has been there eleven years does not follow the discount SOP; she follows the SOP plus forty corrections to it that she has accumulated case by case. Nobody has written the corrections down, because to her they do not feel like rules. They feel like experience. This knowledge leaves the building whenever she does.

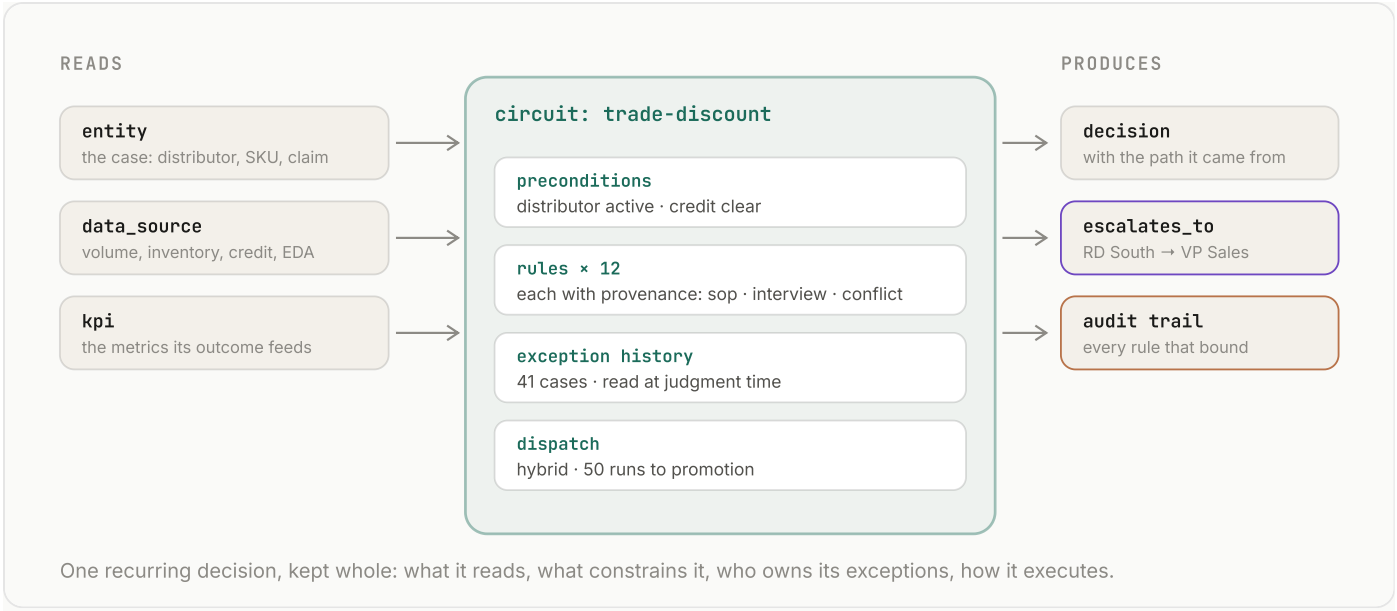
The remainder is visible only in the traces. Some constraints nobody can tell you in advance, in a document or an interview, because nobody knows they hold until a live case runs into them. The record of decisions actually made, with their overrides and escalations, is the only place this layer exists.

Every serious attempt to put AI on company decisions runs into this distribution. A system trained or prompted on the documents alone performs like a smart new hire on their first week: correct on the standard cases and wrong on the ones that matter, because the material that decides hard cases was never written down. The numbers in this paper put that intuition on a footing: in our model, unwritten constraints are **38.3%** of the rulebook and account for **81%** of the failures a documents-only system produces.

The Intelligence Warehouse's answer is to give each recurring decision a home: the decision circuit. A companion paper covers how circuits get used at answer time, when a traversal walks into one. This paper is about the circuit itself: what it holds, where its contents come from, and how it changes over a year of operation.

02 What a decision circuit is

A decision circuit is a typed subgraph that captures one recurring decision in executable form. Not a description of the decision. The decision, in a form that runs.



One recurring decision, kept whole: what it reads, what constrains it, who owns its exceptions, how it executes.

A circuit holds six things.

`preconditions`	What has to be true before this decision applies at all. Checked first, because they are the cheapest checks and they filter out most of the traffic.
`rule nodes`	The codified constraints. Each one carries its provenance as a first-class attribute: which SOP it came from, which interview, which resolved conflict. A rule without an origin does not exist in a circuit.
`escalation path`	Typed `escalates_to` edges to the people who own the exceptions, in order. This is what makes the circuit's output an approval chain rather than a suggestion.
`exception history`	The accumulated record of every case where the rules were overridden, by whom, and why. Read at decision time, so judgment on an unusual case is anchored to how this company handled similar cases, not to a general model's sense of what is reasonable.
`dispatch policy`	How the circuit executes: `deterministic` (compiled code), `agentic` (a model reasons at the judgment point), or `hybrid` (a router chooses per case). Section 10 covers how a circuit moves between these.
`interfaces`	Typed edges to what the circuit reads and affects: the entities it acts on, the data sources it consults, the KPIs its outcomes feed. These are what let a traversal enter the circuit from anywhere in the company graph.

Like every node in the IW graph, a circuit exists at two resolutions. At a traversal frontier it costs about 45 tokens: name, scope, one line. Its full body, rules and exception history included, loads only when a walk actually enters it. A company can hold thousands of circuits, and any given question pays for the two or three it touches.

The word "circuit" is chosen over "skill" deliberately. The two are close cousins: small, self-describing procedures loaded on demand. The difference is origin. A skill is a file someone wrote for an agent. A circuit accumulates out of the company's own operation, and every rule in it points back to the document, the interview, or the resolved case it came from. It is not installed. It grows, and the rest of this paper is about how.

03 The model we measure with

Claims about knowledge capture are easy to make and hard to check, so the paper carries a simulation, and every number from here on comes out of it. The setup matches the companion paper: one company, 420 recurring decisions, 8,000 decision runs a month, traffic Zipf-distributed because a real company's question traffic concentrates on a head of recurring cases.

Each circuit has a true rulebook of 8 to 25 constraints, and each constraint has two properties. The first is **origin**: where it can be learned from.

ORIGIN	SHARE OF RULES	SHARE OF BIND WEIGHT	WHAT IT MEANS
`doc`	61.7%	81.5%	Written in an SOP or policy. The common, agreed constraints.
`interview`	27%	16%	Held by an experienced person. The exceptions and corrections.
`trace`	11.3%	2.5%	Learnable only from live cases. Nobody can state these in advance.

The second property is **bind weight**: how often the constraint actually matters on a live case. Documented rules are the common ones, so they carry high weights. Interview and trace rules bind rarely, but when they bind they decide the case. This asymmetry is the whole game: the rules that are hardest to capture are the ones that separate a right answer from a plausible one.

A decision run samples one to three binding constraints by weight. If any binding constraint is missing from the circuit, or present but stale, the run is an **intervention**: a person has to override, rework, or rescue the outcome. The intervention rate is the quality measure for everything that follows. We also give the world a pulse of genuine novelty: 1.5% of runs hit a constraint nobody has seen before, which no capture mechanism can prevent. That is the floor, and it is why no line in this paper reaches zero.

On top of this, the three capture mechanisms beyond documents (interviews, trace capture, reinforcement) are switches we can turn on and off independently, which is what lets §08 say what each one is worth rather than crediting the whole system at once.

04 Layer 1: documents

Circuits are born from paper. SOPs, policy documents, delegation-of-authority matrices, process wikis: everything the company has bothered to write down gets parsed into draft rule nodes, each tagged with the document and section it came from.

This layer is cheap, fast, and genuinely valuable. It captures roughly **61.7%** of the rulebook, it requires nobody's calendar, and it produces the skeleton every later layer attaches to: the thresholds, the named approvers, the standard cases. In the simulation it is not a switch; it is the floor every world starts from.

It has two limits worth stating plainly. The first is that documents hold what everyone already agrees on. An SOP is the part of a decision that was uncontroversial enough to write down, which means the difficult residue, the part that generates arguments and escalations, is systematically absent from it. The second is staleness: in the model, 6% of documented rules are wrong as written, because the process moved and the document did not, and a wrong rule that the system confidently applies is its own source of failures.

The measured consequence of stopping here: the intervention rate opens at **34.09%** and stays at **33.5%** twelve months later. Nothing improves, because a documents-only system has no way to learn what the documents never contained. Roughly one decision in three keeps needing a person to catch it, forever.

05 Layer 2: expert interviews

The corrections in the pricing analyst's head do not come out through a form. They come out through a conversation with someone who asks the right follow-up questions, and that is a job the IW gives to Morrie, its interviewing agent.

An interview session for a circuit does three things a questionnaire cannot. It walks the expert through live edge cases rather than asking for rules in the abstract, because people who hold tacit knowledge cannot enumerate it but can react to cases all day. It probes disagreements with the written SOP directly, since the places where the expert quietly

deviates from the document are exactly where the undocumented rules sit. And it writes what it learns as rule nodes with the expert's name on the provenance, so six months later nobody wonders where a constraint came from.

Interviews are expensive in the one currency that matters, senior people's time, so the rollout is head-first: circuits get interviewed in traffic order, 60 circuits a month in the simulation, which covers the decisions that carry most of the volume within the first quarter. Each interview captures 85% of that circuit's interviewable rules; nobody remembers everything in one sitting, and the model does not pretend otherwise.

What the layer buys, measured: turning interviews on takes the month-one intervention rate from **34.09%** to **10.4%**, because the head of the traffic gets its exceptions early. The long-run picture is more interesting and less obvious: by month 12, trace capture (§06) would have found many of the same rules the hard way, so the interviews' end-of-year contribution looks smaller in the decomposition than their early-months contribution. The two things interviews buy that nothing else can are time, a year of failures on the head circuits that never happen, and insurance, which is §09: a rule that reaches a circuit before its owner resigns is captured, and one that does not may be gone.

06 Layer 3: decision traces

Once a circuit is live, every run through it is evidence. The trace layer turns the failures into rules.

When a run ends in an intervention, something specific happened: a person looked at the system's output, knew better, and did something else. That correction contains a rule. The trace mechanism investigates the delta between what the circuit concluded and what the person did, drafts the missing constraint, and routes it to the circuit's owner for confirmation. In the simulation, an intervention caused by a missing rule gets that rule codified with probability 0.5; the rest of the time the case was too ambiguous to yield a clean constraint, which matches the experience of anyone who has tried to run a postmortem process.

Two properties make this layer different in kind from the first two. It reaches the rules nobody could have stated in advance, the `trace`-origin constraints that only exist once reality produces the case. And it never stops: documents are ingested once and interviews end, but the trace layer runs as long as the company does, which is why it is the largest single contributor to the month-12 result in §08's decomposition. The write-back

mechanism from the companion paper is this layer seen from the traversal side: a conflict resolved during a walk lands as a rule node, and the next walk through the same territory reads the rule instead of rediscovering the problem.

There is a cost baked into how this layer learns, and it should be said out loud: every rule the trace layer captures was paid for with a real intervention, a case the system got wrong in production. Traces are the cheapest mechanism per rule and the most expensive per lesson. That trade is the strongest practical argument for not skipping the interview layer on any circuit that matters.

07 Layer 4: reinforcement from usage

The first three layers add rules. The fourth judges the rules already there, using the one signal a live platform produces for free: what people do with the system's decisions.

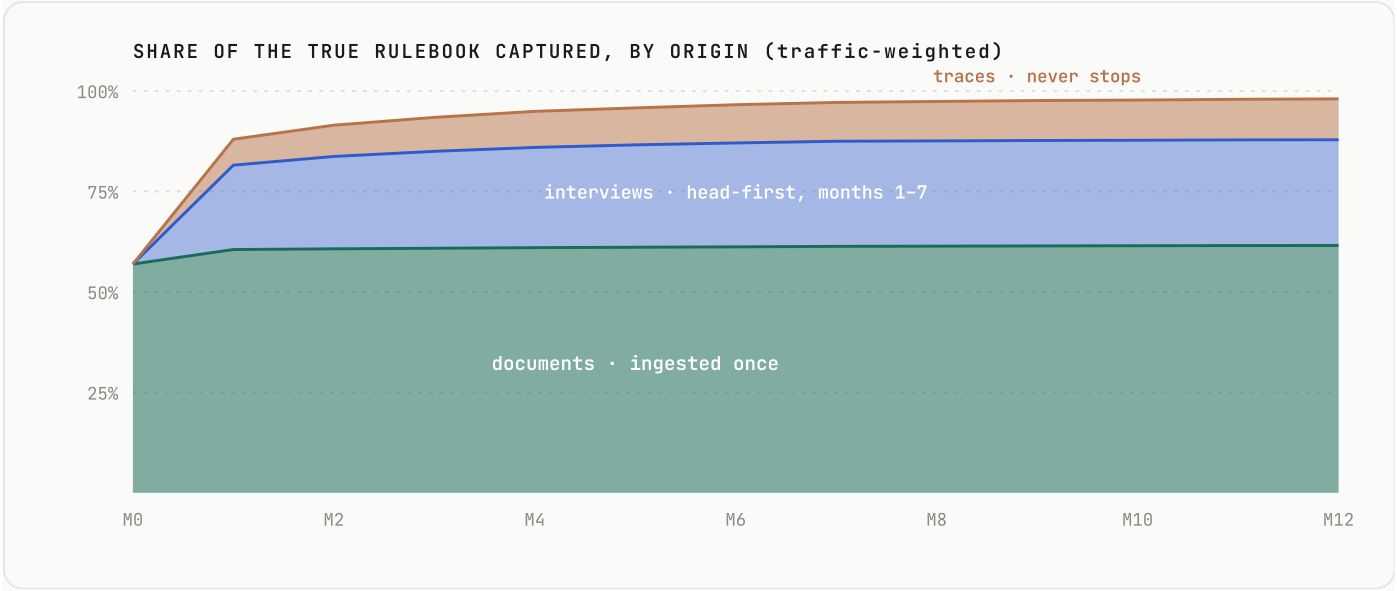
Every run ends in one of three outcomes. The decision was accepted, it was overridden, or it was escalated. Each outcome is a label on every rule that participated in the run. A rule that keeps sitting inside accepted decisions earns confidence. A rule that keeps showing up in overridden ones is either wrong, stale, or missing a sibling exception, and after enough implicating cases (three, in the simulation) it gets flagged and corrected through the circuit's owner.

This is the layer that deals with the 6% of documented rules that were wrong from the day they were ingested, and with rules that rot later as the process underneath them moves. It also feeds the dispatch decisions of §10: the confidence scores that reinforcement accumulates are the evidence a circuit presents when the system proposes promoting it to deterministic code, and drift in those scores is what triggers demotion.

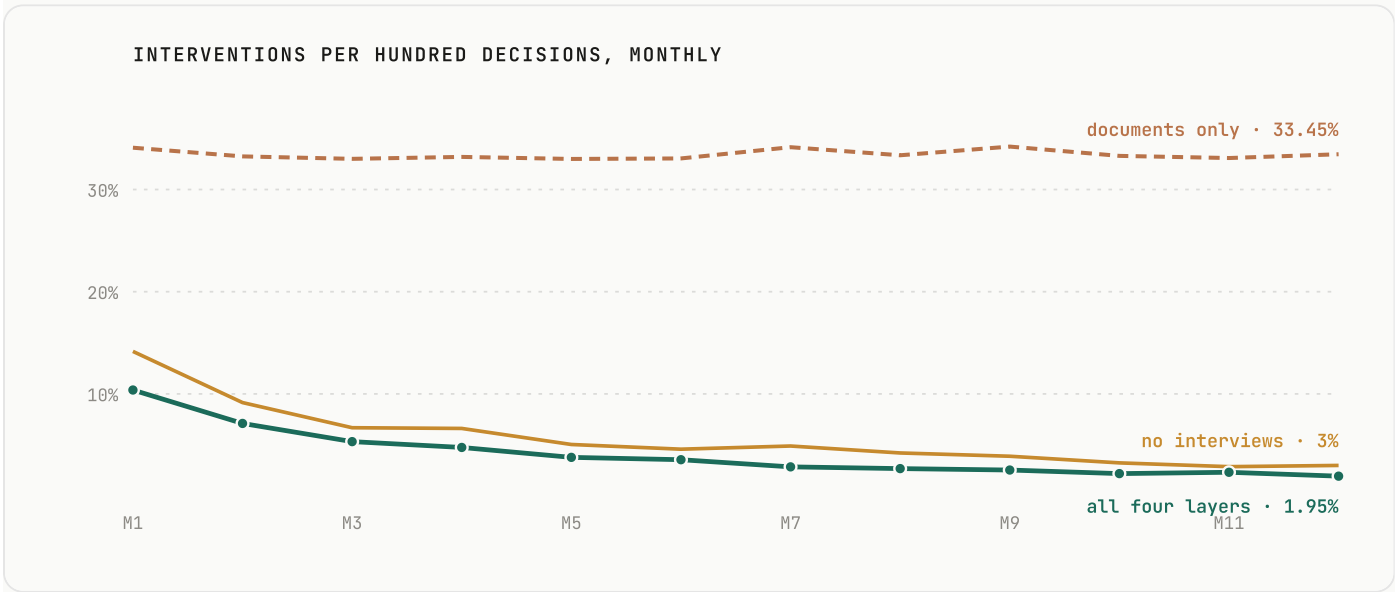
A note on the word, because "reinforcement" invites a stronger reading than we intend. Nothing here updates model weights. This is outcome-weighted bookkeeping over explicit rules: accept and override signals adjusting per-rule confidence, with a human owner in the loop for every correction. It behaves like reinforcement learning in the loose sense that usage improves behaviour, but every update is a legible change to a named rule that someone approved, which is the difference between a system that learns and a system you can audit while it learns.

08 The year, measured

Run all four layers together for twelve months and two curves tell the story: what the circuits know, and how often they still fail.



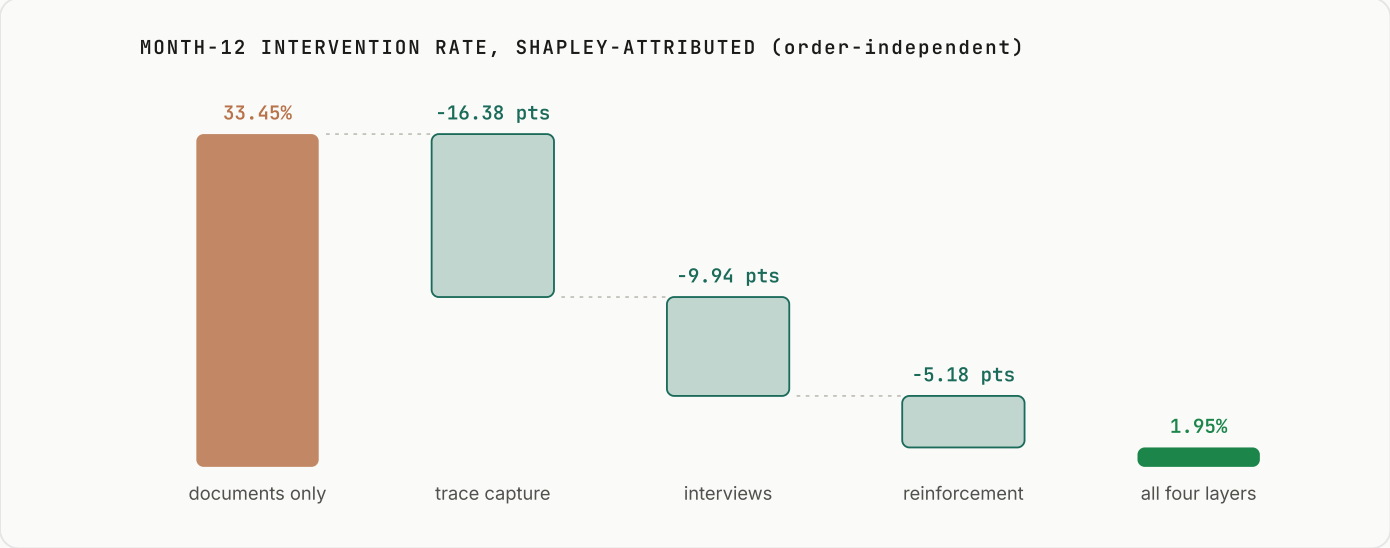
Traffic-weighted share of the true rulebook captured, by origin of the knowledge. Documents provide **56.91%** on day one and nothing after. Interviews add their block in the first months, head-first. The trace layer's slice grows for the whole year and is still growing at the right edge. Total at month 12: **98%**.



Interventions per hundred decisions, monthly. Documents alone hold at **33.5%** indefinitely. The full system opens at **10.4%**, because head-first interviews are already in effect in month one, and reaches **1.9%** by month 12, a **81.2%** decline against its own first month. The middle line drops interviews and keeps the rest; the gap between it and the full line in the early months is what interviews are worth while trace capture is still catching up.

What each mechanism contributed

Because the three mechanisms are independent switches, we ran all eight combinations on the same seed and split the month-12 improvement with Shapley values, so the attribution does not depend on the order you imagine adding them in.



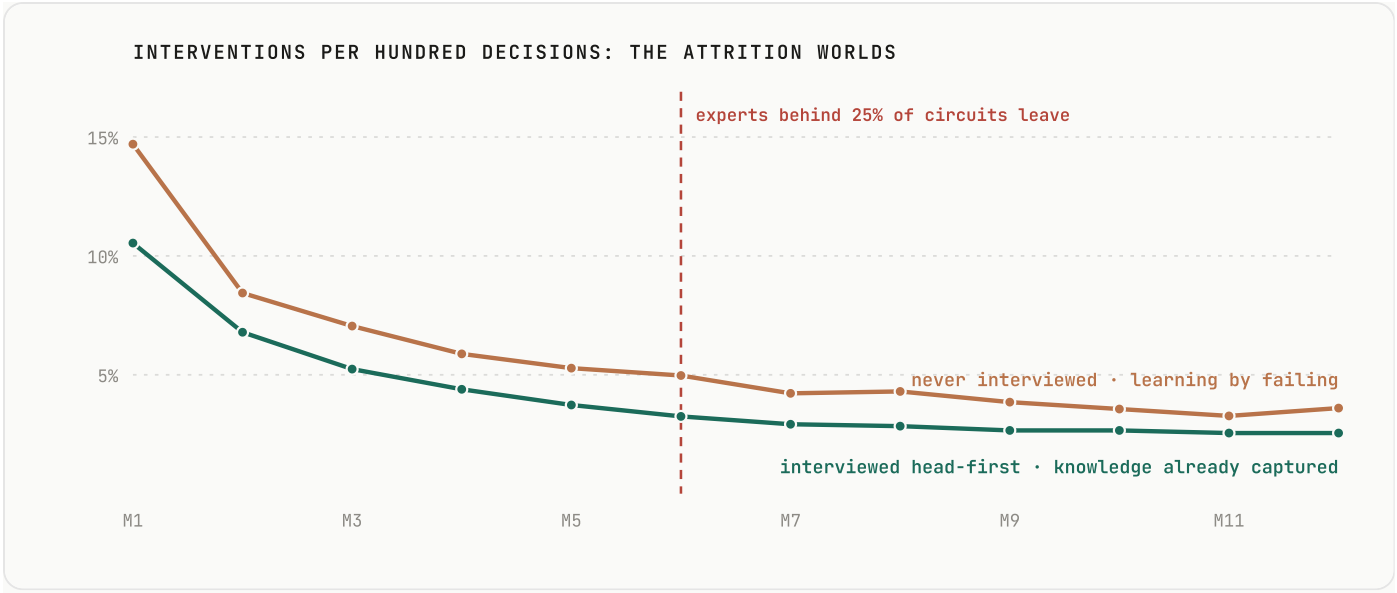
Month-12 intervention rate, from documents only (**33.45 per hundred**) to all mechanisms on (**1.95 per hundred**). Trace capture carries the largest share of the end-of-year figure, interviews the second, reinforcement the remainder.

The decomposition rewards a careful reading. Trace capture wins at month 12 because it never stops running, but the early months belong to the interviews, and a decomposition taken at month 3 would rank them the other way around. The mechanisms are not competing: interviews buy the first year's performance on the decisions that matter most, traces grind down the tail forever, and reinforcement keeps the rulebook honest while both work. Skipping any one of them shows up, just in different places on the calendar.

09 When the expert leaves

Knowledge held by one person is a liability with a resignation date. This experiment prices it.

At the start of month 6, the experts behind a quarter of the circuits leave the company. Their undocumented rules go with them: any of that knowledge not yet captured can now be recovered only slowly and expensively from traces, at a fifth of the normal codification rate, because the person who could have explained the failed case is gone. We run the same year twice: one world that interviewed head-first from month 1, and one world that relied on documents and traces alone.

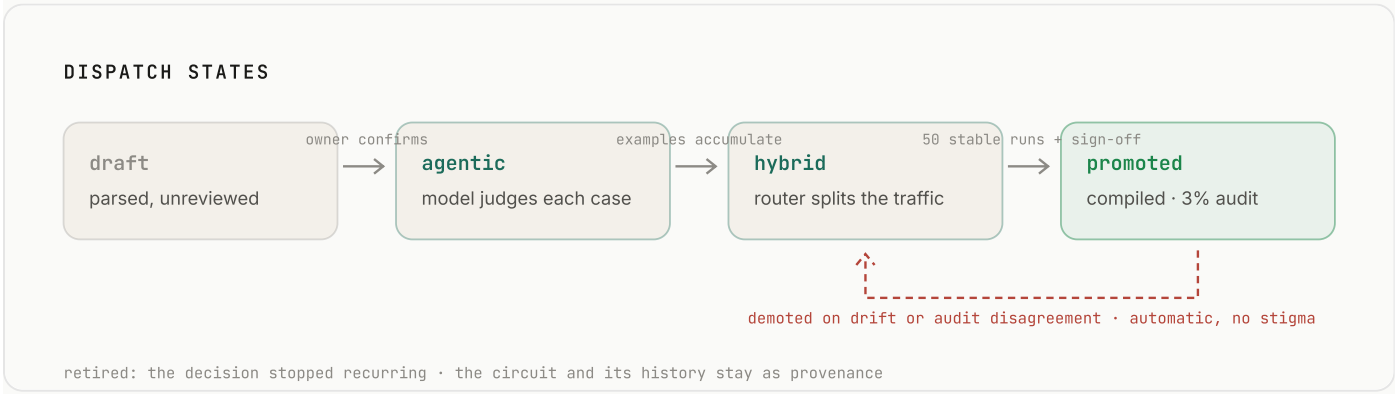


Interventions per hundred decisions in the two worlds. The interviewed world barely notices month 6: what the leavers knew about the high-traffic circuits is already in rule nodes with their names on the provenance. The uninterviewed world runs persistently higher after the departure and never fully closes the gap, because for the affected circuits it is now learning by failing, slowly.

The gap between the two lines understates the real difference in one respect: the simulation only prices the interventions, and in a real company each of those is a wrong or delayed decision with its own cost downstream. The plain reading is the right one. Interviewing a circuit's owner is insurance you can only buy while they still work for you, and the premium is a few hours of their calendar.

10 The life of a circuit

A circuit is not finished when its rules are captured. It has a lifecycle, and the interesting transitions are the ones it makes on its own evidence.



`draft` Parsed from documents, unreviewed. Cannot execute; exists so an owner can confirm the skeleton before it touches live traffic.

`agentic` The normal working state for a young circuit. A model reasons at the judgment point, reading the rules and the exception history. Every run adds a worked example.

`hybrid` A router sends the standard cases down a compiled path and reserves the model for cases that look like exceptions. Most circuits spend most of their life here.

`promoted` Fully deterministic. Reached when the circuit's accumulated evidence shows its judgment has stabilised: in our modelling, fifty consistent runs. A 3% sampled audit keeps checking the compiled logic against fresh model judgment. Promotion needs a human sign-off wherever the circuit writes to a system of record.

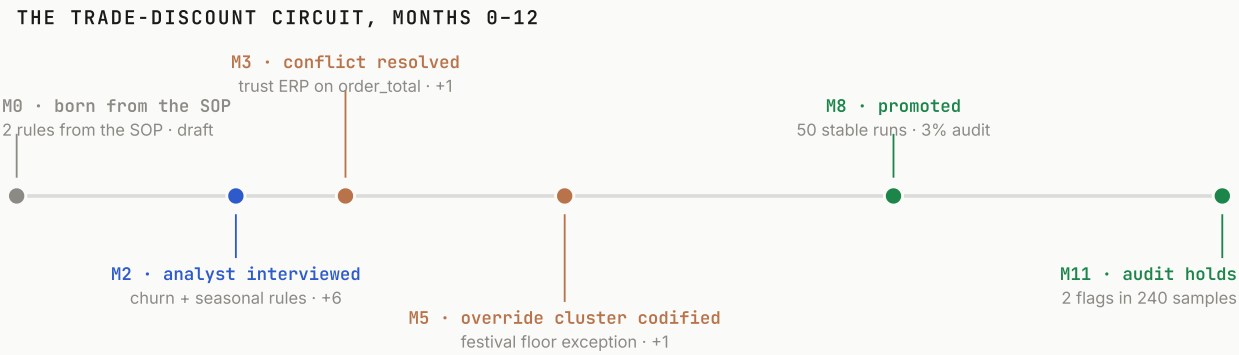
`demoted` Any promoted circuit whose audit disagreements rise or whose input distributions drift goes back to hybrid, automatically. Demotion is cheap and carries no stigma; it is the system noticing the world moved.

`retired` The decision itself stopped recurring. The circuit and its history stay in the graph as provenance for whatever replaced it.

The same move, seen at the company level: a judgment call that keeps coming up eventually gets written into procedure. Companies do this by hand when someone writes an SOP. Circuits do it continuously, with the evidence attached, and in both directions, which is the part companies are bad at: un-writing a procedure when it stops being true.

One circuit's first year

The trade-discount circuit from the companion paper, from birth to promotion:



Every event is one of the four layers doing ordinary work: documents, an interview, two trace captures, then promotion under audit.

Nothing in this timeline is exotic; that is the point. Each event is one of the four layers doing its ordinary work, and at the end of it the company owns, in one inspectable object, a decision that used to live in a PDF, an analyst's head, and a scatter of override emails.

11 What a circuit is not

Every column of this table has been proposed as the place to put a company's decision logic. The rows are what a circuit needs that each alternative lacks.

	SOP DOCUMENT	RULES ENGINE (BRMS)	PROMPT / SKILL FILE	FINE-TUNED MODEL	DECISION CIRCUIT
Executable	no	yes	partially	yes	**yes, with dispatch**
Holds exceptions	rarely	only if enumerated	a few, by hand	implicitly, invisibly	**yes, with history**
Per-rule provenance	n/a	seldom kept	no	no	**mandatory**
Learns from operation	no	no	no	only by retraining	**traces + reinforcement**
Escalation built in	described	no	no	no	**typed edges to people**
Auditable per decision	no	yes	weakly	no	**yes, path-level**
Survives attrition	partly	partly	partly	partly	**that is its job (§09)**

Two of these deserve a sentence. The **rules engine** is the closest ancestor, and its history is instructive: BRMS deployments stall because someone has to enumerate every rule up front and maintain them by hand forever, which is exactly the assumption the four layers remove. The **fine-tuned model** absorbs decision behaviour but cannot show you a rule, cite where one came from, correct one without retraining, or tell you which of its habits just went stale. It is the analyst's head again, only harder to interview.

12 The honest ledger

What circuits cost, and where this analysis could be wrong.

WHAT THE FOUR LAYERS COST

- **Expert time.** Interviews consume hours from the people the company can least spare, and the simulation's 60 circuits a month is a real staffing commitment for the interviewing programme. The head-first ordering exists to make the first hours the most valuable ones.
- **Ownership.** Every circuit needs a named owner who confirms drafted rules, reviews trace-captured ones, and signs promotions. A circuit without an owner decays into exactly the stale document it was built to replace.
- **Interventions as tuition.** The trace layer learns from production failures. Circuits that skip interviews pay for their rules with wrong decisions in the meantime, and §06's caveat applies: cheapest per rule, most expensive per lesson.

WHERE THIS COULD BE WRONG

- **The rulebook model is a model.** The origin mix and bind weights are documented assumptions, not measurements of any company. The sweep in the script varies the undocumented share from 32% to 58% of rules; the docs-only world worsens steadily across that range while the full system's month-12 rate moves only from 2.1 to 2.7 per hundred. The architecture's result is stable; the size of the documents-only gap is company-specific.
- **Interview quality matters less than expected, and that should be checked.** Sweeping interview recall from 60% to 95% barely moves the month-12 rate, because trace capture backfills what interviews miss. What worse interviews cost is paid earlier, as interventions during the backfill, and in the attrition scenario, where there is no backfill. If your traces are thinner than 8,000 runs a month, interviews matter more than this model shows.
- **Codifying a wrong rule is worse than missing one.** The model assumes trace-captured and interview rules are correct once confirmed. In practice confirmation is a human step that can fail, which is why provenance and per-rule reversibility are load-bearing rather than decorative.

13 Reproduce it

Every number on this page comes from one seeded script: the rulebooks, the twelve months, the eight switch combinations, both attrition worlds, and both sweeps.

RUN IT YOURSELF

```
$ python3 iw-decision-circuits-sim.py
# → iw-decision-circuits-data.json    (every figure on this page)
# → iw-decision-circuits-monthly.csv  (the monthly table, per row)
```

sim `iw-decision-circuits-sim.py`: standard-library Python, no dependencies. The rulebook model, the four layers as switches, Shapley attribution over the eight runs, the attrition experiment, and the two sensitivity sweeps.

data `iw-decision-circuits-data.json`: the aggregates this page renders from, embedded below.

seed `20260919`: change it and the draws move; the shape of every curve stays.

The parameters most worth replacing with your own are the origin mix (how much of your decision logic is actually written down; most companies overestimate this), the interview capacity your senior people can sustain, and your monthly decision volume, which sets how fast the trace layer can learn.

INTELLIGENCE WAREHOUSE · RESEARCH · SEPTEMBER 2026

Authors. Akhil Singh, Nidhi Prabhu, and Aniruddha Tammewar.

The claim, in one line. Give each recurring decision a system of record, fill it in four layers (documents for the skeleton, interviews for the exceptions, traces for what nobody could state in advance, reinforcement to keep it honest), and the share of decisions needing human rescue falls from **33.5%** on documents alone to **1.9%** in a year, in an object the company can inspect, audit, and keep after the people who knew it move on.

Figures rendered live from `iw-decision-circuits-data.json`. Simulation seed 20260919 · 420 circuits · 96,000 decision runs per world.