PROFORMA GLOBAL RESEARCH

Learning Instead of Fixing

Matt Rollings, Founder and Principal, Proforma Global

Whitepaper · 2026-05-24

Executive Summary

The pitch for a self-learning agent rarely says this out loud, but it comes down to one promise: the process you never fixed and the data nobody cleaned become the agent's problem now. It will learn the quirks and route around whatever is broken, so the slow and political work of fixing the process can wait. The promise is the trap. An agent left to learn on its own absorbs every workaround it needs, and the workarounds pile up into a system so dense with special cases that breakdown stops being a risk and becomes a matter of time.

Each learned rule is either a genuine, irreducible rule or a fix the organization deferred. The genuine rule is the rare case. The rest is debt the organization chose not to pay down, of three kinds: process, data, and technical.

Some actions do require rules, and that is exactly why autonomous learning is foolhardy. Whether a recurring case is a real rule or a workaround that should have been fixed is a judgment the agent cannot make, and autonomy is the removal of that judgment. The agent encodes the genuine rule and the avoidable one with equal confidence, faster than anyone can review either.

Self-learning makes process design more necessary, not less, because the alternative is a fragile heap of exceptions no one designed, growing until it fails in a way no one can trace. Read correctly, the rules an agent accumulates are a map of the process, data, and technical work the organization still owes.

---

1. The Promise That Process Is Optional

The appeal is real. Process change is the hardest work an organization does. It crosses departments, breaks habits, and shows nothing for months. Cleaning a data source means identifying the owner of the data and making them understand the downstream impact. An agent that simply learns to cope looks free by comparison: it asks no one to redesign anything or confront anyone, and the mess keeps working.

So the agent becomes the place inefficiency goes to hide. A reconciliation that takes three analysts and a week is not redesigned, only handed to the agent as it stands. The feed that arrives missing half its attributes is never repaired. A learned rule fills the gaps instead. Every gap

that should have prompted a fix instead prompts a lesson, and the defect it should have surfaced hardens into a permanent fixture.

The trade is a bad one. What looks like the agent sparing you the cost of fixing the process is really the agent moving that cost somewhere you cannot see, and adding interest.

---

2. What the Agent Is Actually Learning

An agent would rarely need to learn anything in a system whose data was clean and whose processes were complete. It learns a rule when the work in front of it diverges from what the process or the data assumed, closing a gap it was never given a clean way to handle. The volume of what an agent learns measures that divergence.

A managed-services vendor's invoices arrive without the cost center they belong to, because the upstream feed never carried it, and the agent learns to infer the cost center from the vendor and the description. The inference works, and it means the agent now quietly compensates for a feed that should have carried the cost center to begin with. Multiply that across a few hundred learned rules and the store stops being a record of intelligence and becomes an inventory of everything the organization routes around.

An agent that learns quickly in a mature, well-run environment is a warning rather than an achievement, because clean data and complete processes leave it little to learn. The agents that survive in production, as "Self-Learning Architectures for Enterprise Agents" argues, learn almost nothing on their own. Heavy learning is a symptom, not a capability.

---

3. Process, Data, and Technical Debt

"Self-Learning Configuration in Enterprise Agents" draws the line between a genuine rule and a fix as the gate on what enters the store. A genuine rule is the rare case: a relationship so conditional that no fix is cheaper, a distinction the business genuinely makes that nothing upstream can normalize away. There is no root cause to repair, because the rule is the root, and it is worth keeping for good. Everything else the agent learns is a fix it deferred, and the deferred fixes are debt of three kinds.

Process debt. The rule works around a process that is manual, broken, or so special-cased that no clean rule was ever going to fit it. The agent reroutes an approval because the official routing is wrong, or repeats a reclassification each close because the process books the entry to the wrong place first. The rule does the process's job and postpones the work of rebuilding it.

Data debt. The same logical fact arrives in inconsistent forms, with missing attributes or from conflicting sources, and the agent learns to reconcile the variants. That reconciliation is work the data should have done once at the source. Normalizing it there removes the rule and every cousin of it, the structural and vocabulary work "Semantic Layers in Enterprise Agent Systems" describes.

Technical debt. Two systems that should be integrated are not, or a platform cannot do natively what the close requires, and the rule stands in for the missing capability. The agent carries data across a gap that should not exist, or rebuilds what an interface dropped in transit, work that belongs in an integration or a repaired system rather than in a rule that re-bridges it every close.

---

4. The Silent Stopgap

Every operation carries workarounds. What changes with an autonomous agent is that it absorbs them silently, leaving no friction behind. A person who hits the same gap a hundred times generates friction: a complaint, a ticket, a cost a manager can see. An agent simply learns its way past, and the friction never appears, so neither does the signal that a fix is needed. The ticket that used to be filed against the bad feed stops being filed, and its owner closes the backlog item as resolved.

The stopgap becomes invisible at the same moment it becomes permanent. Nothing downstream complains about the bad feed anymore, so no one fixes it, and the broken process it feeds is carried quietly every close instead of rebuilt. The organization sees automation and counts it as progress, while underneath, the debt it should have been paying down is capitalized into the agent's rule store. The cleaner the surface looks, the less anyone is looking at what holds it up.

5. The Fragility of a Thousand One-Offs

Each rule the agent adds is sound on its own. The trouble is what they become together. A designed process handles a whole class of cases by its structure, so removing the cause of a problem makes the whole thing simpler. Learned rules carry no such structure. They handle cases one at a time, and the pile only ever grows.

Rules do not sit quietly side by side. Two can fire on the same case and disagree. The clash resolves cleanly only when a business sets precedence between them deliberately, the discipline "Self-Learning Configuration in Enterprise Agents" describes. An autonomously grown pile never had it. Others leave a gap between them, a case each assumed the other would catch. A rule written for one condition can be silently undone by a second built for a different one, and the number of these collisions climbs faster than the number of rules themselves. A person can still hold a few dozen rules in mind. A few hundred one-offs defeat anyone, and nothing general remains that is true of the system, only the behavior that emerges when hundreds of narrow conditions meet a case none of their authors foresaw.

That is what makes the system fragile in the exact sense of the word. When it fails, the wrong answer is the product of the whole pile rather than any single rule, so no one can trace it. Every rule added raises the odds of such a case and lowers the chance anyone could diagnose it, and the system grows harder to change at the same time, because no one can predict what touching one rule will do to the others it silently depends on. This surfaces in a financial close as a number that is wrong for reasons no one can reconstruct. Pile up enough one-offs and breakdown is no longer a risk, only a matter of time.

6. Why Autonomy Is the Wrong Mechanism

None of this is an argument against rules. Some actions genuinely need one, the irreducible cases from earlier, and a financial system with no encoded rules at all would be a fantasy. The argument is against letting an agent decide, on its own, what becomes a rule.

Creating a rule well requires a judgment the agent cannot make. Faced with a recurring case, someone has to ask whether it is a genuine rule or a workaround for a problem that should be fixed instead, and the answer turns on context the agent does not have: whether the broken

feed can be repaired at all, and what the workaround costs across everything it touches. Autonomy is precisely the removal of that judgment. The agent sees a pattern, confirms it works, and encodes it, with no capacity to ask whether the pattern should have existed.

The usual reply is to gate the autonomy: let the agent absorb the low-risk rules and send the high-impact ones to a person, with a confidence score deciding which is which. It does not help, because a confidence score measures whether a rule works, not whether it should exist. The cost-center inference can be right ninety-nine times in a hundred and still be a workaround for a feed that should have carried the cost center itself. Confidence and the rule-versus-fix judgment are different axes, and no threshold on the first stands in for the second.

A few of the agent's rules are genuinely necessary, and that is the problem: the avoidable ones arrive wearing the same clothes. Autonomy stamps both with equal confidence, faster than any review can sort them. The result is the fragile pile, assembled at machine speed.

The discipline runs opposite to autonomy in the one place it counts. An agent can surface a candidate and the cases behind it. Whether it becomes a rule, or a fix the data, the process, or the system should carry instead, is not the agent's decision to make, the separation between proposing and deciding that "Governance as Middleware" draws. Only a rule kept as a deliberate, priced, time-bound choice belongs in the store. "Self-Learning Architectures for Enterprise Agents" sets the discipline of what an agent should be allowed to learn at all. The judgment here is its sharpest case: deciding what should not become a rule is the entire game.

---

7. The Store as a Map of What to Fix

One workaround tells you little. The set of them maps where the process, data, and technical debt actually sits, and the map is worth more than any single rule on it. It shows where the recurring costs concentrate and which root fixes would retire the most rules at once.

Reviewed on the cadence the close already imposes, the store tells an organization what to fund. One feed, fixed at the source, can remove forty inferred-cost-center rules at a stroke, and the same holds for a single broken process whose repair retires a quarter's worth of reclassifications. The rules name their own remedies, for anyone willing to read them as debt rather than as intelligence.

That fragility is never escaped, only outrun. A mature organization mines the store for the fixes worth making and pays the debt down faster than the agent piles it up, watching the store shrink. Where no one reads it, the store grows instead, until the agent runs mostly on

workarounds whose reasons no one remembers and the deferred costs have compounded past the point where anyone can find them. An organization's rule store already says which of the two it is.

8. Boundary

This paper is a position, not a remediation guide. It does not say how to normalize a data source, rebuild a process, repair a system, or sequence the fixes against everything else competing for the same budget. That work is specific to the system and the organization, and it is engineering rather than the argument made here.

It does not revisit how rules are stored, scoped, or inherited, the subject of "Self-Learning Configuration in Enterprise Agents." Nor does it restate which kinds of thing an agent should be allowed to learn at all, settled in "Self-Learning Architectures for Enterprise Agents." This paper assumes both and asks a narrower question: of the rules that pass those tests, how many are standing in for a fix that should have been made instead.

An autonomous agent that learns is not, for the most part, getting smarter. It is recording the distance between how an organization's systems are meant to work and how they actually do, one rule at a time, and the larger that record grows the closer the whole sits to failing in a way no one can trace. A few of those rules encode something real and belong there. The rest are a bill for the process, data, and technical work the organization deferred, and an agent that pays it quietly, on its own, only lets the debt compound. The work the agent was supposed to make unnecessary is the work that keeps it standing.

DOWNLOAD PDF

Work with Proforma Global

The thinking in this paper is public; the methods that turn it into a working system are not. If it fits a problem your team is working on, that is what we bring to an engagement. Start a conversation: **info@proforma.global**.