Digital Workers at Industrial Scale

The Learning Architecture Behind Autonomous AI Agents

White Paper | May 2026

IFS Loops Platform – Enterprise AI Division

Ravi Bulusu

Abstract

Enterprise AI has a credibility problem. The technology works well enough in demos and controlled pilots; the failure typically comes after deployment, when an autonomous agent encounters the messy complexity of real operations without a structured framework for handling it.

This paper argues that the problem is architectural, not technical, and that the right architecture already exists in a place we rarely look: the way human expertise actually develops. Drawing on the Loops framework, we lay out six design principles for industrial AI agents that mirror the intern-to-expert arc: staged autonomy, a three-layer learning hierarchy, collaborative reinforcement with domain specialists, structured memory, a deliberate split between planning and language generation, and pattern-based determinism.

These aren't theoretical constructs. They're the mechanisms by which an AI agent earns the right to operate autonomously - and the mechanisms by which organizations can trust that it will.

The Architecture in Production

The principles in this paper reflect lessons from active deployments across industrial operations. At KGS, a large-scale gas services operator running the Loops Material Replenishment digital worker, the results illustrate what this architecture delivers at full maturity:

90,000+	**$3M+**
hours reclaimed annually	in annual cost savings

Source: IFS Customer Success · KGS Deployment Data

Contents

1. Value Realization Through Staged Autonomy

The Intern-to-Expert Journey

The reliable path to value from enterprise AI agents is rarely the fastest one. Organizations that push straight to full autonomy tend to accumulate a particular kind of hidden risk: failures that are hard to trace, expensive to fix, and damaging to the credibility of the entire AI program. The root cause is usually not the model; it's the absence of a structured development arc.

> Human expertise does not emerge fully formed. A new hire begins by observing, asking questions, and proposing actions for review. Over time, as their judgment is tested and validated, they are granted more autonomy. AI agents must follow the same arc.

The Loops framework defines three distinct stages of agent maturity:

Figure 1: The three stages of agent maturity: from Co-Pilot to Digital Worker, with autonomy rising as trust is earned.

Stage 1 - Co-Pilot	The agent observes context, generates recommendations, and surfaces relevant information. Every decision requires explicit human approval. This is where trust gets established - users see the agent's reasoning and calibrate their expectations accordingly.
Stage 2 - Co-Worker	The agent executes multi-step tasks end-to-end and presents completed outcomes for review. The human is no longer supervising each step - they're reviewing results. Cycle times compress, and throughput gains become measurable.
Stage 3 - Digital Worker	The agent operates autonomously across full workflows. Human-in-the-Loop (HITL) escalation occurs only when the agent hits a novel situation, its confidence drops below a defined threshold, or a pending action carries risk above a configured ceiling. At this stage, the agent is a peer contributor.

"Having a human in the loop at every critical decision point is incredibly valuable, and you've built that into the design." – Joakim Stolt, CIO, Ependion

The Importance of the Journey

What makes the staged approach valuable isn't that it eventually produces a fully autonomous agent (most frameworks promise that). What's different is what happens along the way. Each stage generates concrete data: which task types the agent handles reliably, which edge cases trip it up, where human judgment is genuinely needed versus where it's simply habit. That data shapes the configuration of the next stage.

Skipping stages doesn't accelerate deployment: it shifts cost to a later, worse point. Failures that surface after an agent has been granted full autonomy are harder to diagnose, harder to remediate, and much harder to explain to stakeholders who trusted the system. The staged journey isn't a formality. It's how an agent earns the right to operate without a safety net.

There's a growing gap between the AI organizations claim they're willing to trust and the systems they've actually deployed. While autonomous, exception-driven AI is widely discussed as the future, most enterprises stop short of operationalizing it. That hesitation isn't a technology readiness issue–it's a failure to build for trust from the start.

2. The Learning Hierarchy: Tools, Skills, and Goals

Walk into most enterprise AI deployments and you'll find the same design mistake: an agent given access to hundreds of APIs and dozens of tools, all equally available at all times. It seems like the right approach: give the agent everything it might need. In practice, it produces the opposite of intelligence. Too much context and LLMs lose precision. Too many options and planning degrades.

Three-Layer Architecture

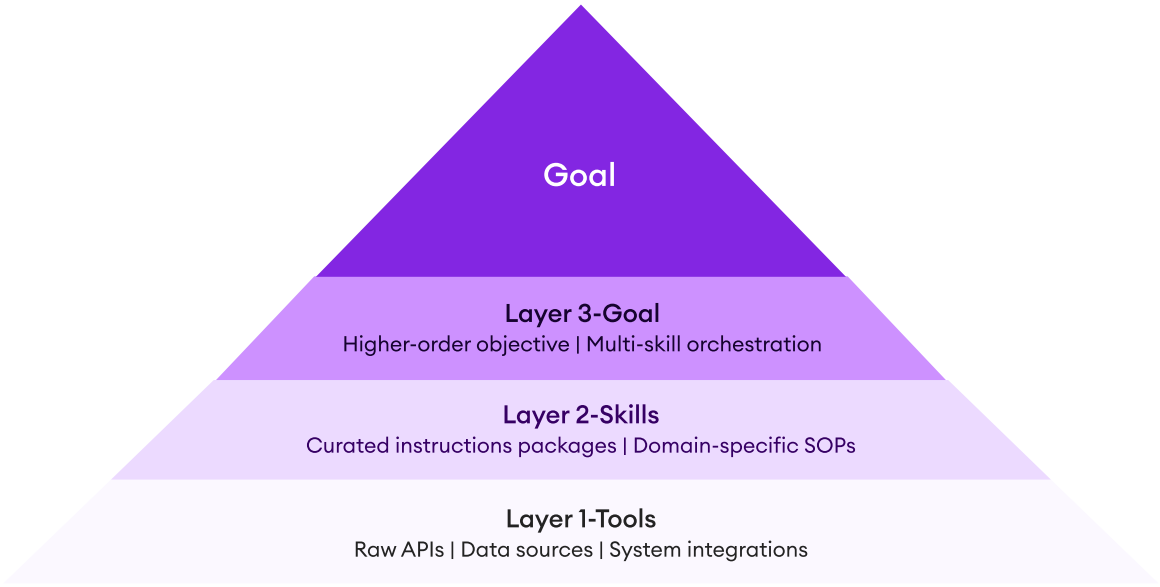


Figure 2: The three-layer architecture: Tools form the foundation, Skills package context, Goals orchestrate outcomes.

Layer 1 - Tools	Raw APIs, data sources, and system integrations. These are atomic capabilities: read a record, update a field, send a notification, query a status. At this layer, the agent knows what operations exist but has no context for when or why to use them.
Layer 2 - Skills	Curated packages of instructions and tool subsets, assembled for a specific purpose. A skill for 'process a warranty claim' might contain six tools, a decision tree, and three conditional branches - and nothing else. Skills dramatically reduce the context burden on the LLM and introduce the first layer of domain specificity. They're the equivalent of a standard operating procedure: not a rigid script, but a structured framework for exercising judgment.
Layer 3 - Goals	Higher-order objectives that require orchestrating multiple skills in sequence, in parallel, or conditionally based on intermediate outcomes. 'Resolve a P2 field service incident' is a goal, not a task. It may invoke skills for diagnostics, parts lookup, scheduling, customer communication, and closure documentation - in an order determined by what the agent learns as it works.

The layered architecture does two things well. It bounds complexity: at any moment the agent works within a defined context rather than scanning an infinite solution space. And it creates a natural development path: skills can be built, tested, and validated in isolation, then assembled into goal chains once they're proven reliable. This modularity matters more than it might seem. It's the difference between debugging a single skill and trying to trace an end-to-end failure with no clear entry point.

3. Collaborative Reinforcement: Learning From Domain Experts

General training only gets an agent so far. A model trained on broad data doesn't know that a specific organization closes warranty claims differently than it handles repair escalations, or that a certain customer segment requires an entirely different communication approach. That knowledge has to come from somewhere - and in the Loops framework, it comes through structured collaboration with the people who already have it.

Two Mechanisms of Domain Learning

The first is coaching. When a domain expert reviews an agent's proposed action and overrides it, that correction isn't simply applied and forgotten. The expert's reasoning (the why behind the decision) becomes training data. The orchestrator model updates to reflect how the expert thinks, not merely what they decided in this instance. Over many such interactions, the agent begins to anticipate how an experienced practitioner would approach a given situation.

The second is continuous reinforcement. Every completed task gets evaluated against real outcomes: did the right action occur, was the objective met, did downstream systems register the expected state change? Successful patterns get strengthened. Patterns that correlate with failures get flagged for review. Neither mechanism requires the expert to be present for every decision - which is precisely the point.

The demand for this capability isn't theoretical. Leaders already recognize how much critical operational knowledge lives only in people's heads and how difficult that makes it to scale, automate, or retain expertise. Coaching isn't a nice-to-have feature or a product differentiator. It is a direct response to a problem organizations know they have and haven't been able to solve any other way.

> Domain experts in the Loops framework are coaches, not supervisors. Supervision requires the expert to be present for every consequential decision, which defeats the purpose of autonomous agents. Coaching builds capability that operates in the expert's absence. The agent learns not just what to do, but why, and that understanding generalizes.

The practical consequence shows up when a senior expert retires or moves on. Their judgment doesn't walk out the door with them. It's already encoded in the agent's planning model, skill library, and memory, captured not in documentation that no one reads but in the behavioral patterns the agent applies every day.

> *"IFS Loops Digital Workers are not just a bolt-on. They are integrated, and are a mix of skills an actual worker is doing every day. My advice is just get started. Once you get started, it is going to learn."* – Jonatan Gustafsson, Supply Chain Manager, Kitron

4. Memory Architecture: Building Durable Institutional Knowledge

Cognitive scientists have long known that human memory isn't a single system. Experts draw on at least three distinct types simultaneously, and each type serves a different function in skilled performance. The Loops memory architecture is organized around the same distinction - in a form that's persistent, queryable, and owned by the organization rather than resident in any individual's head.

The Three Memory Layers

Memory Type	Function	Organizational Value
Episodic Memory	Logs of past task executions, decisions, and outcomes. Enables agents to reference prior context for long-running workflows.	Avoids repeating past mistakes; foundation of pattern learning.
Semantic Memory	Domain knowledge: entities, relationships, business rules, and organizational context. Curated and maintained.	Institutional knowledge that persists regardless of employee turnover.
Process Memory (Working)	Immediate context for the active task: current step, inputs, decisions made so far. Intentionally limited in scope.	Prevents context saturation; keeps LLM inference lean and cost-efficient.

Memory Formation and Consolidation

The first time an agent encounters an unfamiliar situation, it's slow. It reasons through the problem carefully, escalates to a human expert if needed, and records both the reasoning and the outcome. That initial encoding is expensive. But it only happens once; subsequent encounters with similar situations draw on the established pattern rather than repeating the analysis.

Memory consolidates over time through spaced reinforcement, a mechanism cognitive science tells us about durable human learning. Patterns that recur across many task instances get progressively stronger. Patterns that generate corrections get flagged and weakened. The system doesn't just remember; it learns what's worth remembering.

The operational cost of missing this is real and persistent. When institutional knowledge is hard to find, teams are forced to re-solve the same problems again and again, pulling time away from higher-value work. And when experienced employees leave, unrecorded knowledge leaves with them. The episodic memory layer is the architectural response. Each task resolution, each handled exception, and each decision becomes reusable context that the agent retrieves rather than recreates.

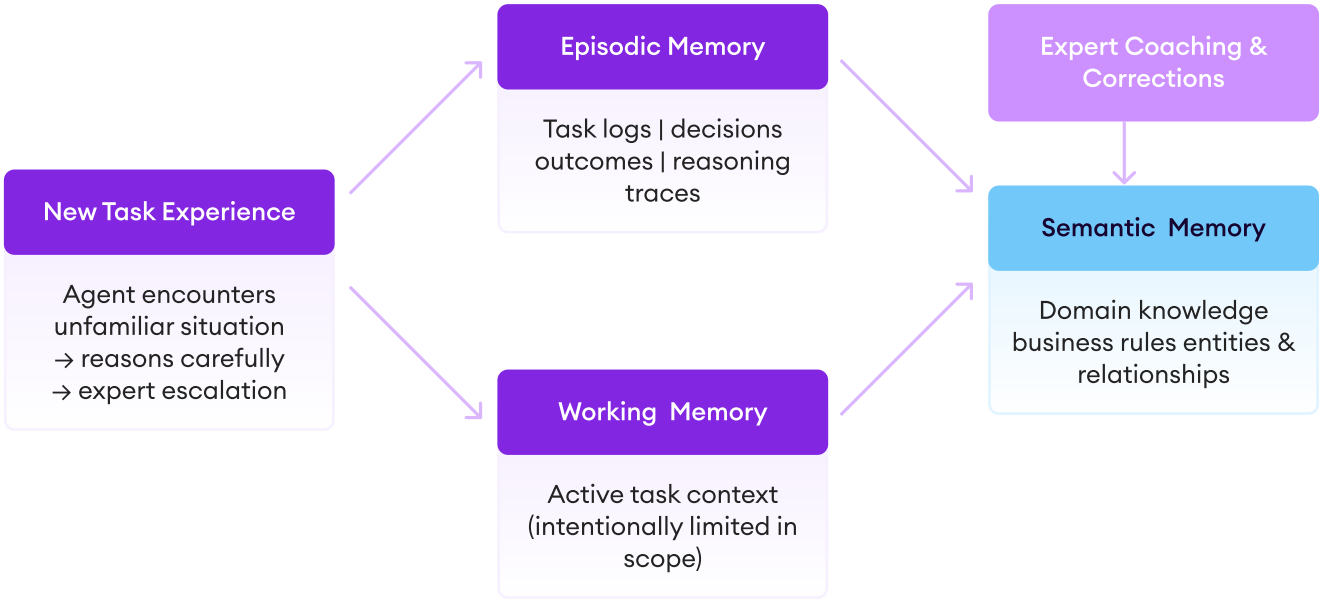


Figure 3: How knowledge flows between memory layers: episodic experience consolidates into semantic memory, expert coaching updates both, working memory retrieves on demand.

Institutional Memory as Organizational Infrastructure

Here's the organizational implication that tends to surprise people: an agent's memory doesn't leave with an employee. When a key practitioner exits, their knowledge doesn't disappear; it's already encoded in the semantic layer, the episodic history, the planning patterns. A new hire, or a newly deployed agent, can be oriented against that accumulated base in a fraction of the time it would otherwise take.

> A second digital worker doesn't start from zero - it inherits the skill library, the semantic memory, and the planning patterns built up by the first. Going from one digital worker to ten doesn't require ten times the investment. It requires the first one to be built well.

5. Planning Intelligence vs. Generative Intelligence

Most enterprise AI architectures make the same design mistake: they ask a single large model to handle both workflow planning and language generation. It seems efficient. In practice, it forces an uncomfortable tradeoff, and the tradeoff almost always goes in the wrong direction for industrial use.

The Cognitive Distinction

Planning Intelligence	Generative Intelligence
Structured reasoning task. Requires understanding current workflow state, selecting from defined actions, sequencing steps correctly given dependencies, and anticipating failure modes.	Language task. Requires understanding natural language inputs, extracting structured information, producing human-quality written outputs, and reasoning about unstructured content.
Benefits from domain specificity, not general fluency.	Benefits from the broad training distribution of large general models.
Must be correct - not merely plausible. Variability is a defect.	Acceptable variability in output - a summary that sounds slightly different each time is fine.
Implemented as: fine-tuned 8–13B parameter orchestrator model	Implemented as: large general-purpose model (OpenAI / Anthropic)

Why Structured Planning Outperforms End-to-End LLM Reasoning

Large general models are remarkably good at sounding right about complex industrial workflows. The outline is often plausible, the language is fluent, the overall direction frequently correct. The failure shows up in the details: steps get sequenced incorrectly given dependencies, available-but-suboptimal tools get selected, and exception conditions that any experienced technician would catch reflexively get missed entirely.

A fine-tuned small language model trained on thousands of planning instances from a specific domain (including the failures) has learned the actual structural patterns of that domain's work. [1][2]

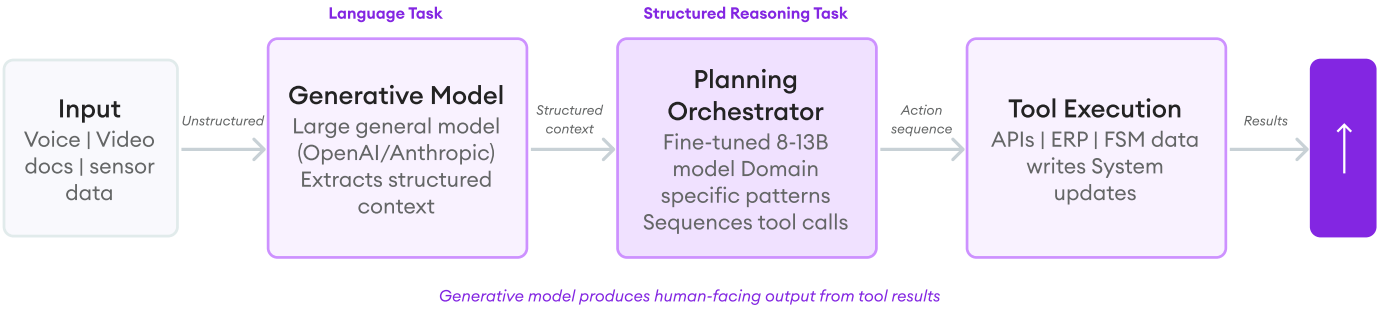
Not a general sense of what workflows look like, but the specific logic of this organization's procedures. Its planning is faster, more consistent, and substantially more reliable. And the generative model, freed from the cognitive burden of planning, does what it's actually good at. The division of labor produces a better system than either component could achieve on its own. [3]

References:

[1] Belcák, P., Heinrich, G., Diao, S., Fu, Y., Dong, X., Muralidharan, S., Lin, Y. C., & Molchanov, P. (2025). Small Language Models are the Future of Agentic AI. arXiv:2506.02153. https://arxiv.org/abs/2506.02153

[2] NVIDIA Developer Blog. (2025). How Small Language Models Are Key to Scalable Agentic AI. https://developer.nvidia.com/blog/how-small-language-models-are-key-to-scalable-agentic-ai/

[3] Patel, C. & Konuk, T. (2025). Llama Nemotron Models Accelerate Agentic AI Workflows with Accuracy and Efficiency. NVIDIA Developer Blog. https://developer.nvidia.com/blog/llama-nemotron-models-accelerate-agentic-ai-workflows-with-accuracy-and-efficiency/



Each model does one thing well | Planning is correct not merely plausible | Language is fluent, not rigid

Figure 4: The dual-model architecture: a fine-tuned orchestrator handles planning; a large general model handles language. Each does one thing well.

6. Pattern-Based Memory and Determinism

How Expert Agents Plan

Ask an experienced field engineer how they diagnose a complex equipment failure and they'll tell you it comes down to pattern recognition. They didn't work through first principles - they recognized the symptom constellation, recalled the response pattern that fits, and adapted it for whatever was specific about this instance. That's what expert cognition actually looks like. It isn't more careful reasoning. It's less reasoning, because so much has already been resolved into pattern.

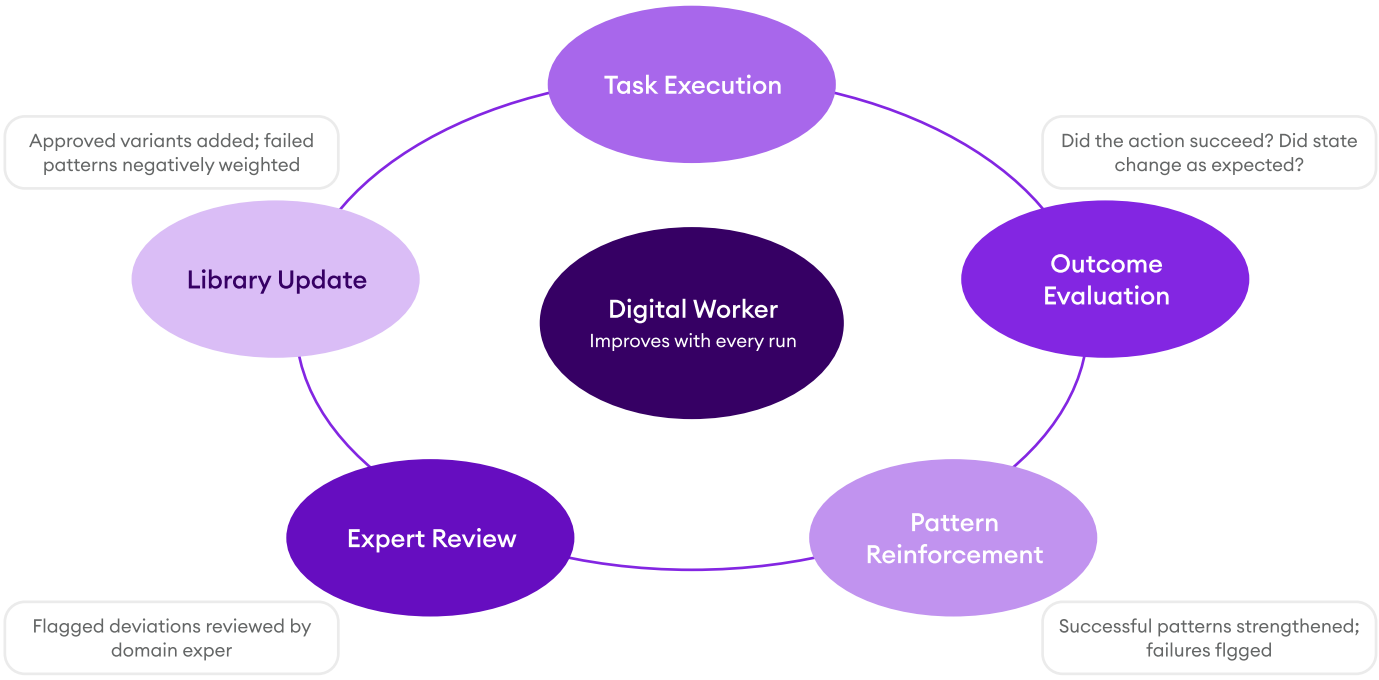
This recognition-action model explains something that often puzzles people: why experienced practitioners are simultaneously faster and more accurate than novices, but also better than rigid rule-based systems at catching anomalies. Speed comes from pattern fluency. Accuracy comes from pattern fluency plus the judgment to notice when something doesn't quite fit.

Pattern-Based Reinforcement in Agent Planning

The Loops orchestrator develops this fluency the same way human experts do: through accumulated experience. Across thousands of planning instances, it learns not just which tools to invoke, but higher-order patterns: action sequences that reliably succeed for a given situation class, recovery pathways for specific exception types, escalation triggers for situations that genuinely need human judgment. These patterns are the product of experience, not programming.

> These patterns are not hard-coded rules; they are probability distributions over planning trajectories, weighted by historical success. The orchestrator approaches a new task not by reasoning from first principles, but by recognizing which known pattern class the current situation most resembles and instantiating that pattern as a starting point. This is the mechanism by which determinism enters an agentic flow.

The Continuous Improvement Loop



Every task execution extends the episodic record that informs future pattern matching

Figure 5: The continuous improvement loop: every task execution feeds back into pattern reinforcement, expert review, and pattern library updates.

Every task execution extends the episodic record that informs future pattern matching. When the orchestrator adapts a base pattern and the adaptation succeeds, that variant gets incorporated into the pattern library. When an adaptation fails, it gets flagged for expert review and, if warranted, added to training data with negative weighting. The system learns from both.

This produces a system that improves with use, not through autonomous self-modification, which would introduce its own risks, but through a governed feedback loop. Significant pattern updates are reviewed before deployment. The improvement is real, but it's supervised. Over time, the digital worker doesn't just perform tasks. It gets better at them.

7. Organizational Implications

The risk is no longer theoretical and increasingly hard to ignore. Many AI initiatives stall before they ever reach full production, and even those that do often struggle to deliver meaningful business value. The breakdown is not in the model itself. It is in the architecture that governs how the model operates once deployed. That is the problem this paper addresses.

Predictable, Auditable Outcomes

Enterprises running autonomous agents at scale can't afford inconsistency in how equivalent situations get handled. The inputs to those situations will always vary: video, voice, sensor data, and unstructured documents are inherently variable, but the decisions shouldn't depend on which day of the week it is or what state the model happens to be in. The Loops architecture addresses this through separation: pattern-based planning enforces consistency at the decision layer, while the generative layer absorbs surface variation in inputs and outputs.

Crucially, the orchestrator's planning decisions are logged, attributable, and tied to identifiable patterns. When something goes wrong, the failure can be located, whether in the planning layer or the generative layer, in a specific pattern variant or a specific tool invocation, and corrected without unwinding the entire system. When failures are tractable, they're fixable. And when they're fixable without rebuilding everything, organizations are willing to deploy more ambitiously.

Scalable Institutional Knowledge

Automation programs typically produce outputs: tasks completed, hours saved, errors reduced. The Loops framework produces something additional: infrastructure. Every request handled, every skill invoked, every expert correction absorbed becomes part of the agent's accumulated knowledge base. The expertise that once lived only in people's heads, unevenly distributed, subject to attrition, and expensive to transfer, gets encoded, structured, and made durable. And unlike most organizational knowledge initiatives, this one compounds. The knowledge base grows more valuable with each deployment, not less.

The Path to Industrial Scale

The staged autonomy model gives organizations a practical, defensible path to scale. Each expansion of autonomous scope is validated against performance data from the stage before it. Trust doesn't get assumed; it gets demonstrated.

The same architecture that supports a single digital worker in one domain scales to a fleet of specialized workers across many. Skills built for one workflow adapt to adjacent ones. The initial investment in skills, memory, and training pays dividends that aren't possible from a series of disconnected one-off automations.

8. Real-World Deployments

The architecture described in this paper isn't hypothetical. The following deployments illustrate what the Loops framework produces in production, across different workflows, different domains, and different stages of agent maturity.

Kodiak Gas Services

Material Replenishment Digital Worker

Before	After
Manual PR review, grouping, and PO creation. High repetition, low tolerance for error, and disproportionate load on experienced planners.	Automated grouping, PO creation, and exception handling. The digital worker manages the full replenishment cycle, escalating only when supplier constraints or inventory anomalies fall outside its learned patterns.

90,000+
hours reclaimed annually

$3M+
in annual cost savings

"This Digital Worker is saving us $3 million in costs, and we are super excited to explore more Digital Workers with IFS Loops." – *CIO, Kodiak Gas Services*

Ependion

Supplier Order Manager
Digital Worker

Before	After
90+ supplier POs processed manually each day from emails and PDFs. Reading, matching, and updating records consumed hours, causing delays, errors, and missed confirmations.	Automated extraction, validation against master data, and exception flagging replaced the manual cycle. Supplier confirmations that once consumed mornings now process in minutes, with discrepancies surfaced immediately rather than discovered downstream.

60%	20 hrs	90+
operational efficiency gain	saved per week	POs processed daily

"Having a human in the loop at every critical decision point is incredibly valuable, and you've built that into the design." – *Joakim Stolt, CIO, Ependion*

KLN Family Brands

Customer Order Manager
Digital Worker

Before	After
2,558 customer POs per month, with 80% requiring manual follow-up and every order reviewed by hand before approval. Buyers spent 80% of their week reading confirmations, identifying discrepancies, and chasing corrections. Delays hit 40% of orders.	Customer POs are extracted, validated against inventory, and confirmed automatically. Discrepancies are flagged at detection. Buyers shifted from data entry and rework to exception management and strategic decisions.

60%	20+	100+
operational efficiency gain	hours saved per week	customer POs processed daily

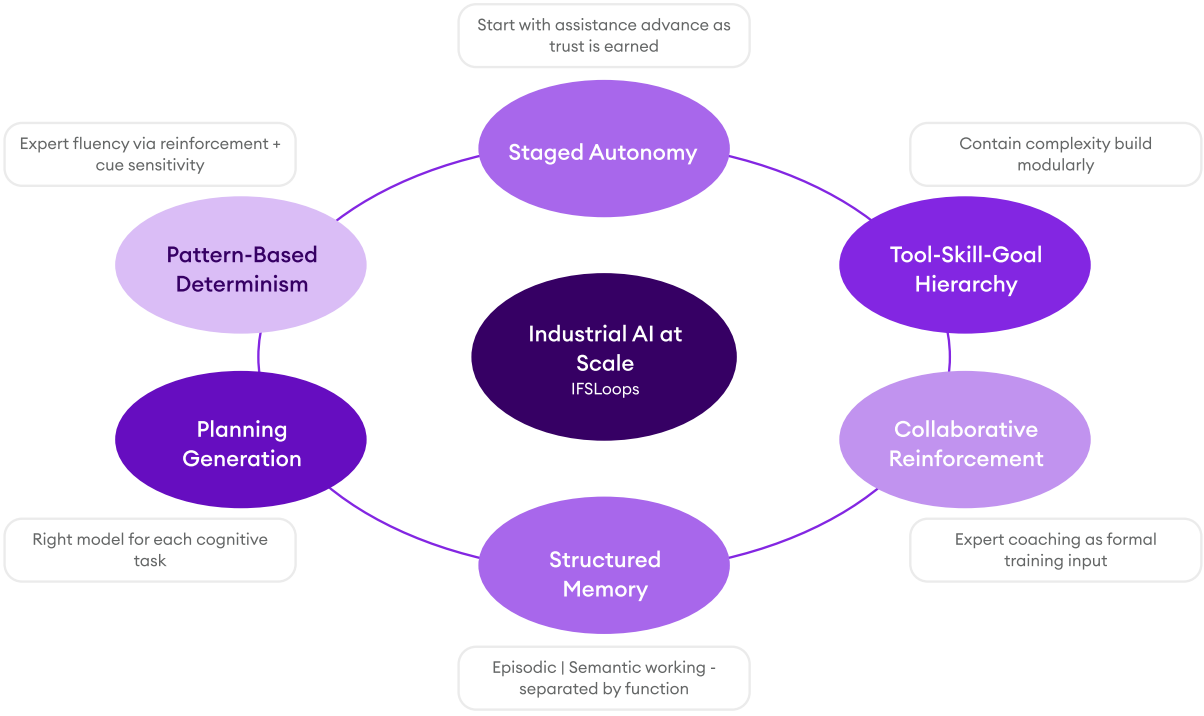
"This is the new agentic world, it moves fast, and it looks super cool. We see many ways to put it to work for us." – *Lance Shultz, IT Director, KLN Family Brands*

Beyond efficiency: where the value shows up

The metrics above tell one part of the story: hours saved, costs reduced, throughput increased. But for most organizations, the more important question is where that capacity goes. When Digital Workers take on repetitive, execution-heavy work, experienced teams shift from operational processing to exception handling, strategic decision-making, and higher-value priorities that were previously deprioritized. The case for Digital Workers is not just doing the same work faster. It is giving your best people their jobs back.

9. Conclusion

Getting autonomous AI agents to work reliably in industrial environments isn't primarily a model capability problem. The models are capable enough. The gap is architectural, and it's addressable with the right design. That design has six interdependent components:



Six interdependent components - removing any one degrades the reliability of the whole

Figure 6: The six interdependent principles of the Loops learning architecture: removing any one degrades the reliability of the whole.

1. **Staged autonomy:** starting with assistance and advancing to autonomous action only as trust is earned through demonstrated reliability.
2. **The tool-skill-goal hierarchy:** containing complexity at each layer, building competence incrementally, and enabling modular extension.
3. **Collaborative reinforcement:** incorporating domain expert coaching as a formal training input, not merely as output approval.
4. **Structured memory:** separating episodic, semantic, and working memory to serve distinct cognitive functions, and building institutional knowledge that outlasts any individual employee.
5. **Planning-generation separation:** assigning structured reasoning and language production to systems optimized for each, eliminating the reliability tradeoff that plagues single-model architectures.
6. **Pattern-based determinism:** achieving expert-level planning fluency through accumulated reinforcement, with the cue sensitivity to adapt when patterns don't quite fit.

The capacity freed by Digital Workers has a clear destination. When operational workload is reduced at scale, organizations do not want to spend that time doing the same work faster. They want to redirect it toward growth, expansion, and new initiatives that have been out of reach. The case for Digital Workers is not efficiency. It is reach, the ability to pursue opportunities the organization cannot today because its best people are too busy executing.

> Agents that follow this path do not merely automate tasks; they develop expertise. They earn trust. They become organizational assets rather than organizational experiments. That is the standard Loops holds its digital workers to. And it is the standard industrial organizations should demand from any AI deployment that operates at scale.