The Work Outlives the Agents

By: Steve Mahoney, SVP of Product, Legion Intelligence

AI has delivered extraordinary technological breakthroughs, but I'm not yet convinced we're having equally extraordinary product breakthroughs. We've got models that can reason through complex problems, write software, analyze enormous amounts of information, and take actions on our behalf. We've given those models tools, memory, and the ability to coordinate with other agents. For all that progress, however, much of what we're calling AI innovation in enterprise software looks remarkably familiar.

We're adding assistants and copilots to existing applications, building agent managers, agent builders, and dashboards to monitor what those agents are doing, and creating increasingly sophisticated ways for people to instruct, coordinate, and supervise AI. These are meaningful advances, and in many cases they make the products they're attached to better. But I'm concerned we're making the same mistake we've made during previous technology "transformations": focusing on the technology rather than rethinking the work it should improve.

People don't wake up and go to work thinking about managing agents. They go to work to accomplish something, like launching a product, closing a new hire, or getting a failed pump back online.

Even a broken pump gets an agent

Adding agents to every application still leaves the supervisor carrying the work between them.

Imagine for a moment that you're a maintenance supervisor at an industrial facility and a critical pump has just failed. You probably open the asset management system, look through equipment records and maintenance history, check for open work orders, and see which replacement parts might be in inventory.

It might feel like a far departure from agentic lawyers and AI bots thirsty for your credit card, but even asset management systems are adding AI agents and chatbots. They can retrieve technical manuals, summarize previous repairs, generate work orders, or recommend

replacement parts. These capabilities are useful, and if you've ever clicked around in a 40-year-old asset management system, you'll welcome the help.

The supervisor's job, however, is to restore reliable operation safely and with minimal disruption. And the information needed to do that extends well beyond the asset management system. Technician availability lives in workforce scheduling. In Microsoft Teams, a tech might mention that repeated seal failures could be caused by equipment misalignment. In a meeting, the team might establish that the backup pump needs to be running before repairs begin. Someone still has to weigh the financial and operational consequences of getting the repair wrong.

Even with agents embedded in all these applications, the supervisor may still be collecting information, evaluating its implications, coordinating people, and monitoring progress. The agents make individual tasks easier, but the human remains the connective tissue between systems.

Now imagine software designed around the actual work objective: restoring and maintaining reliable pump operation. It assembles the relevant information across those systems and conversations, connects the technician's observation to the equipment's repair history, and recognizes a plausible alignment problem. Instead of displaying another collection of records, it frames the decision: replace the seal immediately, or investigate alignment first? It presents the evidence, assumptions, uncertainty, alternatives, and likely consequences so the supervisor can decide.

The software then coordinates the authorized work, updates the relevant systems, monitors progress, and adapts as new information arrives. And it understands that closing the work order isn't necessarily the same as restoring the pump to reliable service, the same test Edge AI Capability Has to Be Proven in the Application applies to fielded agents. What makes this product different isn't the number of agents it uses. It's that the software understands and advances the work, while keeping the person accountable for the outcome in the driver's seat.

We've seen this movie before

The companies that won the machine learning era used it to make difficult work simpler, and the same test applies to agents.

When data science and machine learning became mainstream in the late 2000s and early 2010s, seemingly every software company needed an ML strategy. Products added predictive

analytics, recommendation engines, and elaborate demonstrations of the latest technology. Even Apple found itself in a smartphone machine-learning arms race with Google and Samsung, showcasing the capabilities appearing in familiar products.

Some of those features fundamentally improved products. But the more interesting companies of that era used the technology to reinvent the experiences they offered. Netflix rethought how people discovered entertainment. Uber built a real-time marketplace where matching riders and drivers, estimating arrival times, and responding to supply and demand became the product experience. Instacart created a simple consumer experience above an extraordinarily complicated set of interactions between inventory, shoppers, retailers, substitutions, and delivery logistics.

Those companies didn't win customers by asking them to become data scientists. They built products that made difficult experiences dramatically simpler, using intelligence to do the complex work behind the scenes. The opportunity came from rethinking the product, not making the technology the product.

I think the opportunity is even bigger this time. Data science gave software powerful ways to predict, personalize, and optimize within relatively well-defined environments. The companies above designed the entities, interactions, and constraints of their services, and machine learning made those systems smarter. Increasingly capable AI can now interpret ambiguous objectives, reason across unfamiliar information, use tools, generate plans, and adapt as circumstances change. This opens up work that conventional software has struggled to handle: work crossing applications and functions, dependent on conversations and institutional memory, with steps that aren't always known in advance.

That's much of everyday organizational work. For the hundreds of applications an organization might deploy, humans have effectively been the glue: translating, coordinating, and deciding what happens between them. The data science revolution changed what software could predict and optimize. The agentic software revolution should fundamentally change how organizations accomplish work.

Agentic software development shows the promise... and the limits

Coding agents are the best case for agentic AI, and they still leave people to verify, review, and ship the result.

One of the clearest examples of AI already delivering product value is agentic software development. Coding agents can take a relatively direct expression of intent and turn it into an artifact. They inspect code, make changes, run tests, identify problems, and iterate. There's a natural relationship between what the user wants, what the AI produces and, importantly, how the result can be evaluated.

But even here, execution is messy. In 2025, a Replit coding agent deleted a live database despite instructions to freeze changes [1]. In Stack Overflow's 2025 Developer Survey, developers identified nearly correct AI-generated solutions and time-consuming debugging as common frustrations [2]. The problems go beyond producing code: understanding the project's context, coordinating reviews, and verifying that a change is genuinely ready for use.

This is still an important glimpse of the future. Developers care whether the software works and that the change solves the problem, not which agent wrote a particular function and not how many lines of code were produced. And code comes with something many professions don't have: tests that can help establish whether an output works. For a maintenance supervisor, intelligence analyst, or business leader, success often depends on decisions, people, and changing conditions across several systems. Producing an artifact is only part of getting the work done.

The work outlives the agents

A repair, an investigation, or a launch runs for days or months, long after any single agent or conversation has ended.

Much of today's AI software is organized around interactions. Someone starts a conversation, instructs an agent, receives a result, and moves on. More sophisticated implementations coordinate multiple agents or execute longer workflows, but the unit of interaction is often still the task or the agent responsible for it.

Real work doesn't operate that way. An equipment repair may last days and it happens in the real world. An investigation may last weeks. A product launch can span months, involve dozens of people, and change direction repeatedly. Agents complete tasks, people change roles, new evidence appears, and priorities evolve. The work continues.

That means software needs to maintain something more durable than an agent's memory or a conversation history. It needs to preserve the objective, current progress, decisions already made, the evidence behind them, constraints that must be respected, unresolved questions,

and the relationships connecting it all. It needs to understand what changed, what those changes mean, and what should happen next. One agent might investigate a problem, another might develop a plan, and a third might coordinate execution. A person might intervene and change the objective altogether. None of those transitions should require the work to start over, which is the coordination problem Earning a Seat on the Staff describes inside military staffs.

An agent can finish a task. The software must understand whether the work is finished and, importantly, whether it was successful. That's a different starting point for product design.

From systems of record to systems of work

Each system of record holds a piece of the job, and none of them owns the objective.

Enterprise software has largely been organized around systems of record for the past few decades. We have platforms for managing employees, equipment, customers, inventory, financial transactions, and countless other business entities. These systems serve important purposes: they preserve authoritative information, enforce constraints, and maintain the operational records organizations depend on. But their boundaries don’t mirror the boundaries of the work people need to accomplish.

That pump repair might involve maintenance, scheduling, procurement, and operations. Launching a software product can require product management, engineering, design, marketing, and sales. Just ask a senior product manager how many systems they need access to in order to do their job. The objective doesn't belong to any one of those applications.

I believe truly agentic software creates the opportunity for us to move beyond systems of record and rethink what a system of work can be. A system of work is a persistent, intelligent system organized around what an organization is trying to accomplish. It understands objectives, maintains the state of the work, coordinates people and agents, supports decisions, adapts its approach as circumstances change, and actively moves work toward its intended outcome.

This is more than a conventional workflow engine with an AI interface. Formal processes exist for good reasons, and safety, security, and accountability require rules that must be followed. But much of everyday work is messy: people encounter exceptions, discover evidence, seek advice, and change priorities in ways nobody thought to model beforehand. Rather than prescribing every possible sequence, people can establish objectives and boundaries while

the software determines and adapts how it executes. Predictable steps should still be executed predictably. When the evidence or information is incomplete or there's insufficient authority, the system should involve the right people. Autonomy can grow with trust and reliability.

Importantly, a system of work doesn't require replacing existing systems of record. Those systems can remain authoritative for the information and operations they manage. What changes is the center of gravity: the software experience is organized around the work and its intended outcome, rather than whichever application stores a particular record.

A system of work also doesn't have to map to a department or span an entire organization. Consider a company validating a new product idea. That objective might draw on product, engineering, design, marketing, customer conversations, and go-to-market teams. The people involved may participate in several other systems of work at the same time. A maintenance system might likewise contribute to a broader operational objective alongside procurement and production. These systems can overlap and collaborate while retaining their own responsibilities and understanding. We don't need one enormous system that knows everything; we need systems that can understand work within their scope and cooperate when objectives intersect.

The future of software is one where its boundaries increasingly follow the work, rather than forcing work to be broken up across the boundaries of really smart software.

The next breakthrough is a product breakthrough

Agent and task counts are the new vanity metrics, and they say nothing about whether the work got done.

Organizing software around work also changes the experience people should expect. Today, we either build predefined interfaces for every role or ask people to assemble dashboards and applications themselves, a gap The Next No-Code Platform Is Natural Language takes on directly. But the same pump failure calls for different decisions from a technician, supervisor, and executive. They should be able to engage with a shared understanding of the work through the evidence, comparisons, approvals, and controls appropriate to their role and need in the moment. That doesn't mean endlessly reinventing the interface; familiar design patterns and predictable interactions still matter. The interface is where human judgment meets intelligent execution.

There's a longer-term opportunity as well. A system that moves and preserves work across time can learn from the evidence, decisions, and outcomes that work produces. It might notice recurring failures, identify a pattern that no one has formally documented, and recommend preventive work before the next breakdown. As intelligence improves, systems of work could help organizations discover better ways of operating and even propose objectives worth pursuing. The work itself becomes a source of organizational intelligence.

Ultimately, the measures of success need to shift, too. How many agents ran? How many tasks did they complete? Those are just the new vanity metrics. We need to measure decisions and how much we've reduced humans coordinating between systems. Sometimes an objective can't be met, and progress means recognizing the obstacle, explaining it, and proposing a change in direction. Great software should make those outcomes visible and help people act on them.

This is the bet we've made at Legion Intelligence. We design and build the applications deployed on our platform around the job to be done, not around the records a system keeps. We call these Legion Packs. Each one is centered around the objective of the team, the agents and tools that serve it, enforces approvals by a person before anything consequential runs, and it maintains the state of the work as people and agents come and go.

We know that the person closest to the work, the operators and doers, will always know their work best. Legion's mission builder lets those folks describe it in plain language and assemble it from approved parts, in whatever environment the work lives on, whether classified, disconnected or both. At a U.S. Department of Energy national laboratory, we built a drafting tool for work planning documents on this model, with role-based review and staged approval, and each approved plan becomes a reference the agent checks the next draft against.

Adding agents to existing applications is a useful step, and often a responsible one. But it's clearly not the destination. We have an opportunity to revisit compromises that shaped enterprise software for decades: humans coordinating across applications, workflows modeled in advance, interfaces fixed to a particular role, and organizational knowledge reconstructed by hand. As intelligence grows, those become product-design opportunities rather than unavoidable constraints.

The AI industry's next great breakthrough may come from making models more powerful. But the breakthrough that changes how most people work will come from building products that finally know what to do with that intelligence.

References

1. Beatrice Nolan. An AI-powered coding tool wiped out a software company's database, then apologized for a 'catastrophic failure on my part'. Fortune, July 23, 2025.
2. Stack Overflow. AI, 2025 Stack Overflow Developer Survey. 2025.

Continue the series

Follow Legion Intelligence on LinkedIn: linkedin.com/company/legion-intel

Explore all Command Papers: legionintel.com/command-papers

Ready to talk: legionintel.com/contact