# The Agent Gap

The Coming AI Workload Your Business Can't Handle (Yet)

Arcade.dev

# Executive Summary

For many enterprises, the first wave of AI showed up as conversation. Systems retrieved information, summarized content, or generated responses, but execution still depended on people navigating enterprise systems. That handoff is where work continued to stall.

AI agents change that dynamic. Instead of stopping at insight, they are increasingly capable of taking action — updating records, coordinating workflows, and carrying tasks forward the moment intent is expressed. Early pilots across sales, support, operations, and internal services are already showing real value, and growing pressure to demonstrate ROI is accelerating this shift from conversation to action.

As organizations try to scale these deployments, a familiar pattern emerges. What works well in demos and small pilots becomes brittle in production. Issues around permissions, governance, reliability, and coordination surface quickly, slowing progress and creating uncertainty.

This paper argues that these challenges are not the result of immature models or poor execution. They reflect a structural mismatch. Most enterprise systems were designed for environments where humans always resolve intent before action. Identity models, permission frameworks, and workflows assume people — not software — are the actors doing work.

As agents begin to participate directly in execution, those assumptions break down.

This whitepaper provides a framework for understanding that moment. It explains why action-taking agents place fundamentally different demands on enterprise systems, why existing architectures struggle to support them at scale, and what kinds of architectural questions leaders should be asking before making long-term platform decisions. It is not an implementation guide. It is a foundation for clearer thinking about readiness, risk, and what comes next.

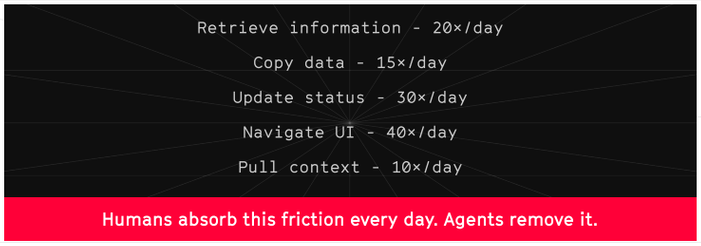
# The Work Around the Work

Before the workday begins, the systems are already in motion.

The customer conversation you flagged yesterday is summarized and ready for follow-up.
The CRM fields you touched are surfaced with suggested updates.

Support tickets arrive with relevant context already attached.
Your morning meetings have the right documents and notes assembled.

By the time you sit down, the small but essential pieces of coordination are already in place — the work around the work.




None of this is unfamiliar. Enterprises have long relied on systems that assist with routine steps: CI pipelines that deploy code, automation tools that route tasks, and integration platforms that keep data in sync. These systems do not replace human judgment. They reduce the coordination overhead that surrounds it.

AI agents extend this pattern in a natural way. Instead of navigating multiple interfaces or remembering how each system behaves, people express what needs to happen, and software carries out the steps across applications on their behalf.

With the right foundation, this kind of support fades into the background. The enterprise feels easier to operate — more responsive, more fluid, and more aligned with how people actually work. The agent era is not about work happening without people. It is about reducing the distance between intention and execution.

# From Conversational Systems to Action-Taking Systems: Why This Moment Feels Different

The scenarios above reveal a boundary that earlier generations of enterprise AI rarely crossed.

Once software begins to participate directly in work, it stops being loosely coupled to enterprise systems. It must invoke tools, coordinate applications, maintain state over time, and carry tasks forward rather than simply produce outputs for people to act on. That shift changes the role software plays inside the enterprise.

Across the industry, AI systems are now being designed to operate inside workflows in this way.
These action-taking systems are what most teams now refer to as AI agents.

| Conversation Systems | Action-Taking Systems |
|---|---|
| Retrieve information | Take actions |
| Summarize content | Update system state |
| Answer questions | Execute multi-step flows |
| Short interactions | Longer-lived tasks |
| Output = text | **Output = real work** |

This distinction matters because it changes what enterprise systems are being asked to support. Systems built for conversational systems can remain loosely integrated – exposing data, accepting queries, and returning responses. Systems supporting action-taking systems must do more, enforcing permissions, exposing reliable operations, handling partial failure, and operating safely across applications.

At small scale, this difference is easy to miss. Early agent demos often feel smooth, with tasks completing automatically and coordination happening out of sight. As action-taking systems expand across workflows, users, and systems, however, the environment around them begins to matter more than the systems themselves.

These systems do not fail quietly. They surface inconsistencies, brittle integrations, and unclear boundaries. They amplify whatever structure — or lack of structure — already exists.

This is why the current moment feels different from previous waves of enterprise AI. The value is real. The gains are tangible. But many organizations are discovering that the systems underneath are not yet designed to support this mode of operation.

# The Hidden Barrier: Enterprise Systems Weren't Designed for Agents

Enterprise systems were built around a simple assumption: a human is always the one taking the action.

Identity models, permission frameworks, approval flows, and audit trails all presume a person navigating an interface, interpreting context, and resolving ambiguity along the way. That assumption is deeply embedded across enterprise architecture, and for decades it has held.

As software begins operating inside workflows, however, those assumptions start to strain.

The challenge is not that agents are unpredictable or immature. It is that enterprise systems lack a native way to represent software acting on behalf of a person, with appropriate boundaries, accountability, and context. What appears at first as friction is actually a deeper mismatch between how work now moves and how systems were designed to govern it.

That mismatch surfaces first in two places.

**1. Identity Was Built for Humans**
Agents do not have a natural place in enterprise identity systems. They do not log in, hold credentials, or map cleanly to permission models designed around people and roles.

When software acts on behalf of a user, enterprise systems struggle to answer basic questions: who is actually taking the action, which permissions apply, and how the action should be audited or attributed. These questions are fundamental, and without clear answers, teams fall back on workarounds.

Early agent deployments often rely on shared credentials, oversized service accounts, or loosely scoped API keys. These approaches allow agents to function, but they come at a cost. Security boundaries blur. Accountability weakens. Governance becomes harder to reason about as agents expand across systems and users.

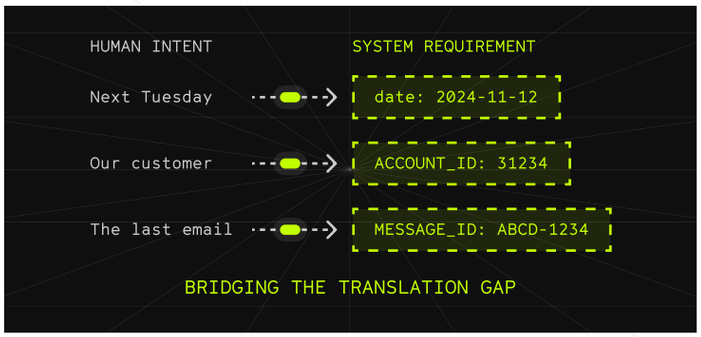
The issue is not agent behavior. It is that enterprise identity systems were never designed to express delegated action by software acting for a human.

**2. Enterprise Systems Expect Deterministic Inputs — Agents Operate From Intent**
The second mismatch is subtler, but just as consequential.

Historically, enterprise platforms were designed so that interfaces forced intent to be resolved before any action occurred. If someone wanted to log a meeting in a CRM, the interface required a date. If they wanted to update a record, they had to select specific fields and values. The burden of translating intent into structured input lived with the human, enforced by the interface itself.

Agent-based systems invert that model. Instead of navigating forms and required fields, people now express intent directly: "log my meeting," "update the opportunity," "move this forward." The interface no longer forces structure up front. That responsibility shifts into software.

The underlying enterprise systems, however, have not changed. They assume that intent is fully resolved before execution, and are built to operate on precise, validated, deterministic inputs. As intent resolution moves out of human-operated interfaces and into software, agents inherit that responsibility — translating fuzzy, contextual intent into the rigid structures those systems require.




This is where the mismatch emerges. Systems designed for fully specified actions are now being driven by software operating from unresolved intent.

The agent is not confused.
The environment is mismatched.

Enterprise systems were built for perfect inputs. Human intent is anything but.

## Why This Breaks at Scale

In early pilots, these mismatches are often manageable. Workflows are narrow, scope is limited, and outcomes are closely monitored. When something breaks, a human steps in to correct it, often before downstream impact is felt.

Many of these pilots are also intentionally designed for a single user or a tightly controlled test account. Identity is implicit or hard-coded. Permissions are hand-scoped. Governance is largely manual. Under these conditions, architectural gaps remain mostly invisible, and systems appear more robust than they actually are.

As deployments expand, those assumptions begin to collapse. Agents are expected to operate across more workflows, on behalf of more users, and over longer periods of time. Identity can no longer be implied. Permissions can no longer be static. Governance can no longer rely on human oversight. The margin for manual intervention disappears, and small inconsistencies start to propagate.

**What was once a localized issue becomes a feasibility problem.**

Consider a simple example: an agent tasked with updating opportunity records. In a pilot, it may act for a single salesperson with carefully scoped access. Errors are visible and easily corrected. As that same agent moves toward production, however, the central question shifts. It is no longer whether the agent can perform the task, but whether the system can be defended at all.

> Who is the agent acting for?
> Under what authority?
> With what limits?

Without clear answers, the project stalls—not because the value is unclear, but because no one can credibly sign off on the risk. This is the moment many organizations are approaching now. Early success gives way to growing fragility. Confidence turns into caution. Security teams raise concerns.
Executives hesitate—not because the promise of agents is unconvincing, but because the foundations no longer feel solid.
At this stage, the limiting factor is no longer model capability. It is architectural readiness.

## The Missing Layer

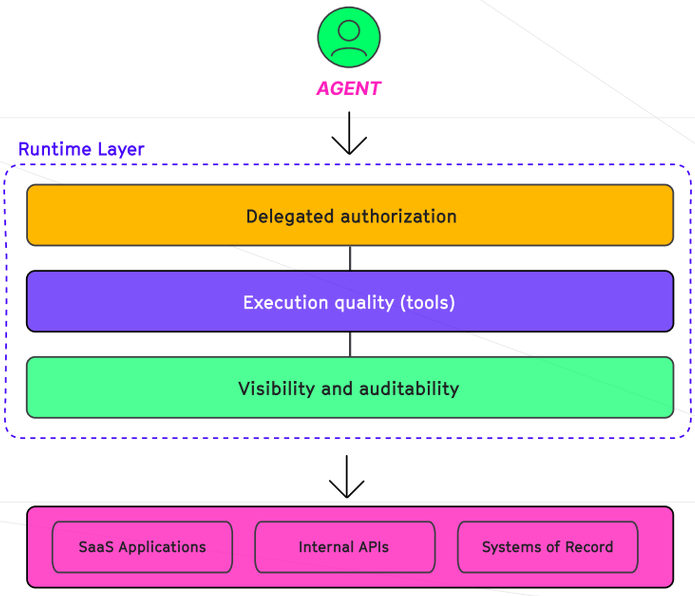
What makes this moment difficult is not agent capability, but the absence of a place for agents to operate safely inside existing enterprise architecture.

As software begins to participate directly in work—updating records, coordinating systems, and executing multi-step tasks—it inherits responsibilities that were previously absorbed by interfaces and humans. Identity must be delegated. Intent must be translated into concrete action. Execution must be governed and auditable. These responsibilities were never first-class concerns of enterprise systems because, historically, humans resolved them before any action occurred.

That structure works as long as intent is resolved before execution. It has no native place for software that must interpret intent, act on behalf of users, and operate across systems with the same reliability and accountability expected of people.

Agents introduce a new requirement.

To function at scale, they need a layer between intent and execution—one that can carry the responsibilities interfaces and humans previously handled. This layer translates intent into structured actions, applies delegated permissions consistently, and provides visibility as work progresses across systems.




This layer does not replace existing systems.
It allows agents to operate within them without forcing those systems to change how they express permissions, enforce operations, or manage state. In effect, it makes explicit the responsibilities that were previously implicit—handled by user interfaces, manual processes, and human judgment.

Making this layer explicit surfaces a small number of foundational requirements that must be addressed before agents can operate reliably at scale.
These requirements fall into three categories: identity, execution quality, and visibility.

## Acting on Behalf of Users

Agents cannot require new identities.
They require a way to act as a user, with the same permissions, boundaries, and accountability that already exist within the organization.

By extending existing identity foundations rather than bypassing them, agents can operate without shared credentials or over-privileged service accounts.
Actions remain attributable. Audit trails remain intact.
Control stays where enterprises expect it to be.

## Making Actions Predictable

For agents to operate reliably, actions must be expressed in a structured way. Systems need to know what operation is being performed, what inputs are required, and how success or failure is defined.

Tools provide this boundary. They do not attempt to model entire workflows. They define the deterministic building blocks workflows depend on, allowing agents to translate intent into execution without introducing ambiguity at the system edge.

## Providing Visibility and Control

As agents participate directly in work, enterprises need clear visibility into what happened, on whose behalf, and with what outcome.

This execution layer needs to provide that visibility by default. It makes agent behavior controllable and governable in the same way other critical systems are, without adding manual oversight or new bureaucracy.

With this foundation in place, agents stop feeling brittle. They become predictable participants in enterprise workflows, capable of acting at scale without undermining security, reliability, or trust.

This is the difference between agents as experiments and agents as infrastructure.

# Drive Adoption through Adaptation

The shift toward agents does not fail because organizations lack ambition or technical talent.
It falters when existing structures are asked to support a new kind of participant in work. As intent resolution moves into software and agents begin to act inside real systems, adoption becomes less about experimentation in pockets and more about whether the organization can adapt its architecture, controls, and expectations accordingly.

Successful adoption tends to follow a consistent pattern. Literacy comes before scale. Teams begin using agents effectively once they can define scoped use cases, fit them into existing workflows, and provide checks to establish trust. Early momentum often starts bottom-up, driven by individuals solving concrete coordination problems, before scaling more broadly as leadership gains confidence.

Architecture plays a critical role in that transition. It's what can scale adoption safely, while still enabling all end-users to experiment. Trust forms not because agents are impressive, but because their behavior is understandable, attributable, and consistent with existing enterprise controls.

Organizations should not reinvent themselves to accommodate agents. They should adapt by extending familiar structures to support a new mode of execution.

## Learnings from Enterprises

The agent era is arriving faster than many organizations expected. Preparing for it does not require a reorganization, a platform overhaul, or a multi-year transformation. It requires clarity about what actually becomes fragile once software begins to participate directly in work.

For leaders seeing early momentum with agents, the question is less whether to expand pilots and more whether the enterprise is structurally ready to support action-taking software at scale. The difference between progress and pause often comes down to a small set of architectural realities that are easy to overlook early and difficult to correct later.

**1. Focus on High-Friction Coordination**
Agents tend to deliver the most value when applied to high-frequency coordination work rather than positioned as broad, generalized AI initiatives. The most effective early efforts stay anchored in workflows where execution is slowed by interface navigation rather than decision-making, and where small inefficiencies compound across teams. These are the places where the shift from conversation to action becomes tangible — and where architectural limits surface quickly.

**2. Treat Identity and Action as Foundational Constraints**
As agents move beyond demonstrations, two questions become unavoidable: who is this agent acting for, and what is it allowed to do?

At scale, reliability depends less on agent intelligence than on whether the enterprise can clearly express delegated identity and define a bounded set of permitted actions. Without that foundation, teams are pushed toward brittle workarounds that undermine trust, complicate governance, and stall progress. Leaders do not need to design these mechanisms themselves, but they do need to recognize them as first-order architectural concerns rather than implementation details to be deferred.

**3. Insist on a Coherent Path From Prototype to Production**
Most organizations reach a familiar inflection point: a handful of promising agent workflows and growing uncertainty about how to run them safely in production.

What matters at that moment is not adding more pilots, but establishing whether there is a defensible path forward — one that aligns agent actions to user identity, makes execution predictable, and provides visibility as usage grows. That distinction determines whether agents remain isolated experiments or become part of how the enterprise actually operates.

# What Comes Next

The shift toward agents is already underway. Once organizations experience work that moves forward the moment intent is expressed, it becomes difficult to imagine returning to the old model – the value is immediate and the structural implications surface just as quickly.

This paper has focused on explaining why those implications emerge. They reflect a predictable mismatch between systems designed for humans to resolve intent before action and software that now participates directly in work.

For organizations ready to take the next step, the natural question becomes what secure, production-grade deployment actually looks like. How does identity extend cleanly to agents? How are actions governed across systems? How is reliability maintained when real data and real workflows are involved?

The companion whitepaper explores these architectural patterns in detail and outlines a practical path from experimentation to scale. If this paper explains why the enterprise needs a new layer, the next explains how to put it in place.

Arcade.dev is the industry's first MCP runtime enabling AI to take secure, real-world actions. As the MCP runtime, Arcade is uniquely able to deliver secure agent authorization, high-accuracy tools, and centralized governance. Arcade helps teams at some of the largest organizations deploy multi-user AI agents that take actions across any system with granular permissions and complete visibility—no complex infrastructure required.

Learn more and try it for free at www.arcade.dev.

