AN EXECUTIVE PERSPECTIVE FOR REGULATED INDUSTRIES

AI-Native Product Delivery

How regulated enterprises are accelerating innovation while maintaining governance.

> **For senior leaders** in insurance and financial services who are reshaping how their organizations build, test, operate, and govern software in the AI-native era — without compromising on security, compliance, or stability.

VerticalServe Inc. · contact@verticalserve.com

verticalserve.com

EXECUTIVE SUMMARY

The shift is bigger than cloud, mobile, or digital transformation.

Artificial Intelligence is creating a shift more profound than cloud computing, mobile applications, or the digital transformation initiatives of the past decade. For the first time, business users can independently create workflows, applications, automations, reports, and agents — without waiting for traditional software development cycles.

This creates both an unprecedented opportunity and a significant challenge for regulated enterprises. **The opportunity is faster innovation. The challenge is maintaining governance, security, compliance, and operational stability while enabling thousands of employees to become builders.**

Many organizations are responding by deploying AI tools. The leading enterprises, however, are discovering that the real transformation is not about tools — it is about operating models.

THE OLD QUESTION

"How do we use AI?"

THE NEW QUESTION

"How do we redesign product delivery for an AI-native world?"

THE AI-NATIVE OPERATING MODEL IN ONE SENTENCE

AI Product Pods that ship working solutions in weeks, against structured specifications that keep code, tests, docs, and agents aligned by construction — all wrapped in embedded enterprise guardrails that apply to every build, regardless of which AI tool was used.

PODS

Small cross-functional teams · weeks-long delivery cycle · business SME, builder, engineer, QA & product owner

SPECS

Single structured definition of intent · drives code, tests, documentation and agents from one source

GUARDRAILS

Security, data access, automated testing, audit and ownership · embedded in the platform, not bolted on

The remainder of this paper unpacks each of these three components and the decisions a CIO must sequence to adopt them.

1 · The Traditional Delivery Model Is Under Pressure

For decades, enterprise software delivery followed a predictable pattern — Business Request → Business Analysis → Requirements → Architecture → Development → Testing → UAT → Production. This model was optimized for stability, control, and predictability, but not for speed. In many regulated enterprises, even relatively simple business solutions require months to reach production.

INDUSTRY DATA

Enterprise application delivery cycles in regulated industries — from requirements through production release — have averaged **3 to 9 months** and remained stubbornly stable for the past decade, even as cloud, DevOps, and platform investments matured.

Standish CHAOS reports · DORA State of DevOps · McKinsey enterprise delivery benchmarks.

Meanwhile, business teams are increasingly experimenting with tools such as **Claude, ChatGPT, Microsoft Copilot, GitHub Copilot,** and **Copilot Studio** — creating solutions in days rather than months. This growing gap between idea generation and enterprise delivery is forcing organizations to rethink the traditional operating model.

2 · The CIO's AI Dilemma

For the first time, business users can produce working solutions faster than IT can deliver them through the traditional cycle. This creates a strategic question that almost every CIO in a regulated enterprise is now facing:

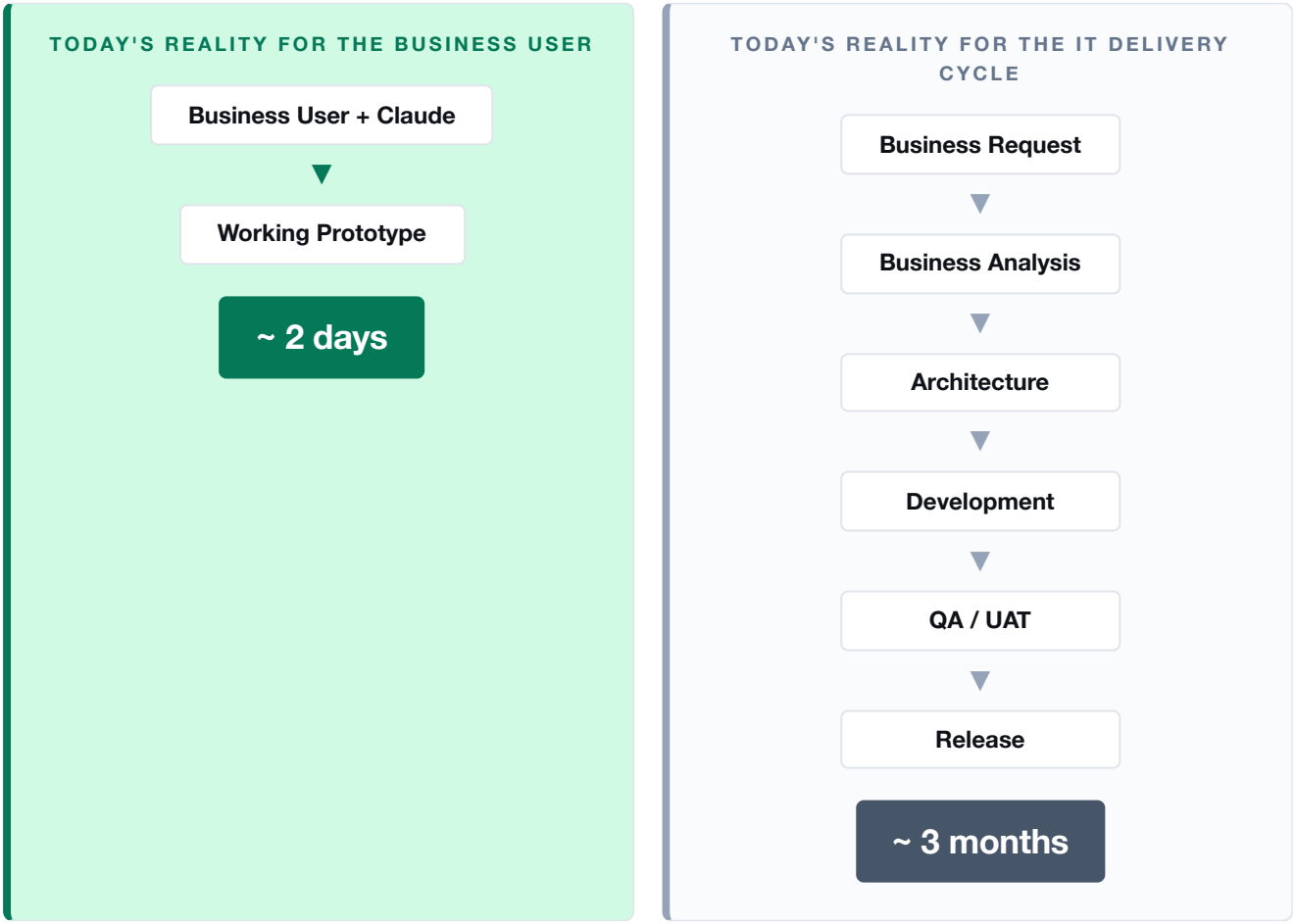
If business users can build useful solutions in days, what becomes the role of IT?

The answer is not to slow business users down. Nor is it to step back. It is to recognize that the disruption is not technological — the same models, the same chips, the same cloud are available to everyone. The disruption is to the *operating model*.

INDUSTRY DATA

Multiple 2024–2025 studies — including **Stanford's AI Index**, **MIT-Sloan**, and **Bain & Company** — report that **well over half of knowledge workers** now use AI tools in daily work. A substantial share of that usage occurs outside formally sanctioned enterprise programs.

Why IT Feels Slower Than Business



THE HARD TRUTH

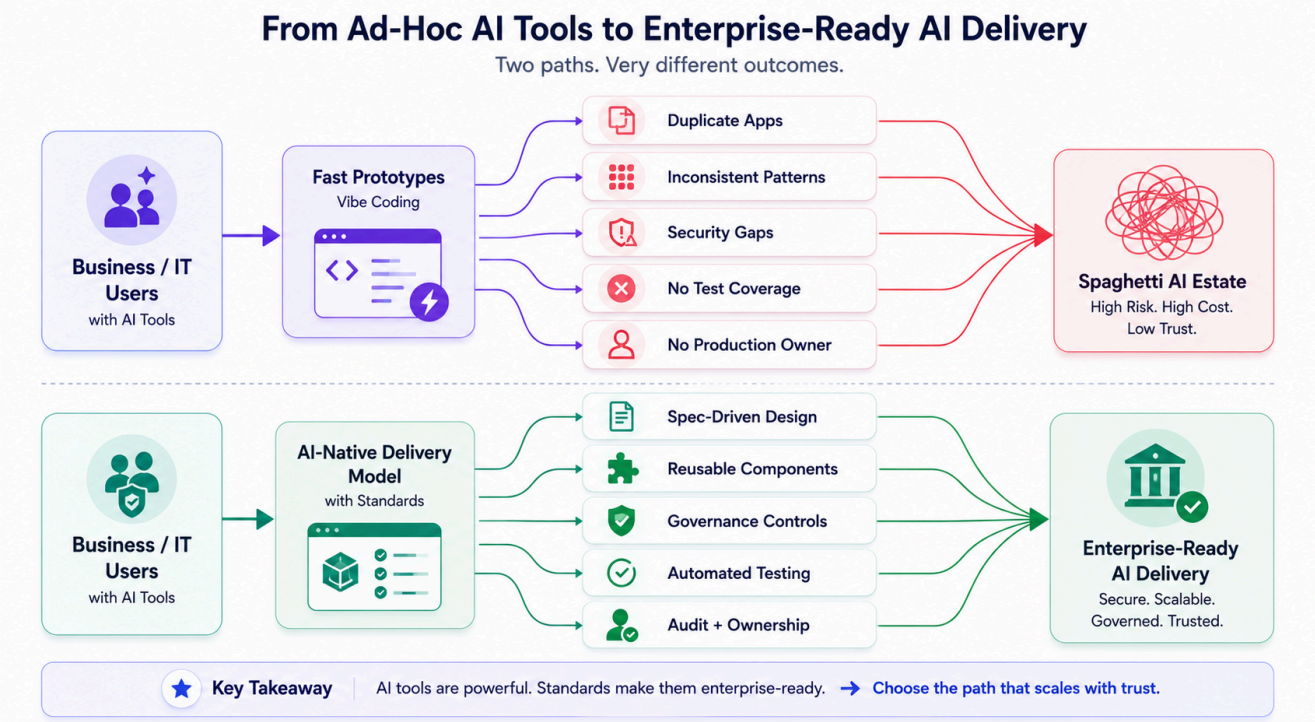
The Problem Is Not Technology.
The Problem Is Operating Model.

The CIO's task is therefore not to acquire more AI tools, but to redesign the operating model so that business velocity and enterprise governance reinforce each other rather than work against each other. The remaining sections of this paper describe what that new operating model looks like.

3 · Vibe Coding Is a Starting Point, Not an Enterprise Operating Model

Vibe coding — the practice of using AI tools to rapidly turn an idea into a working prototype — is useful because it collapses the distance between intention and demonstration. A business user with Claude or Copilot can show a stakeholder a functioning application before lunch. That capability is genuinely new and genuinely valuable.

However, in a regulated enterprise, **speed alone is not enough**. If every team builds independently with different prompts, patterns, models, connectors, and deployment practices, the organization quickly creates a new form of shadow IT: *a spaghetti AI estate* — duplicate apps, inconsistent patterns, security gaps, untested logic, and no clear production ownership.



Two paths from the same starting point. Without standards, speed compounds into risk. With standards, speed compounds into capability.

The answer is not to stop business users or developers from using AI. **The answer is to wrap AI-assisted building with enterprise standards** — spec-driven design, reusable patterns, security controls, automated testing, auditability, production ownership, and governance.

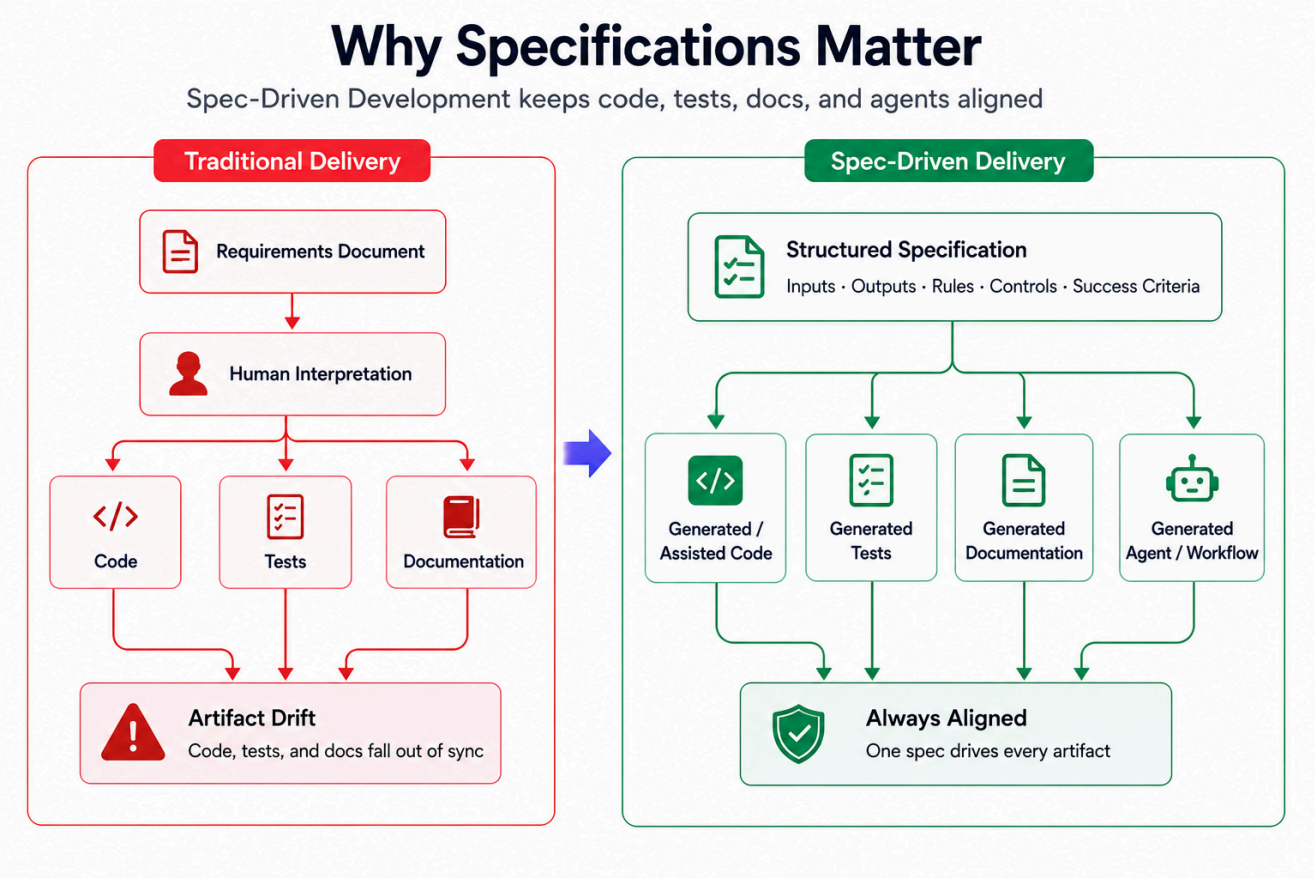
From Vibe Coding to Spec-Driven Delivery

In the AI-native operating model, vibe coding is not eliminated — it is repositioned. It becomes the *front end of innovation*: where ideas are explored and feasibility validated. The remaining steps — design, security, build, test, release, reuse — wrap the prototype with the standards that make it production-ready. Each step is itself AI-accelerated; together they form the eight-stage path from idea to reusable enterprise pattern.

Eight stages — each AI-accelerated, each governed. Vibe coding earns its place as stage 2; the remaining stages translate prototypes into production-ready, reusable enterprise patterns.

Why Specifications Matter

Stage 3 — Spec-Driven Design — is the inflection point of the entire flow. In the traditional model, a requirements document is interpreted by a human into three separate artifacts (code, tests, documentation) which immediately begin drifting apart. In the AI-native model, a single structured specification becomes the source from which every artifact is generated — and re-generated, in lockstep, whenever the spec changes.



Specifications matter because they become the single source of truth. One specification drives the code, the tests, the documentation, and the agent or workflow — and all four stay aligned by construction, not by discipline.

Enterprise Guardrails Around Builders

The pattern works because the standards are not optional and not separate from the tools — they are **embedded**. Builders use familiar AI tools (Claude, Copilot, Cursor, Copilot Studio, and enterprise platforms such as InsightWorker); the guardrails apply automatically. The result is production-ready AI apps and agents that are consistent, secure, tested, and governed by design.

Six embedded guardrails — applied to every build regardless of which AI tool was used — turn AI-accelerated prototypes into apps and agents trusted in production.

In this model, **vibe coding becomes the front end of innovation, but spec-driven delivery becomes the path to production.** The organization captures the speed of the new tools without paying the price in inconsistency or risk.

4 · AI Changes Who Can Build

Historically, business users identified requirements and IT built solutions. AI fundamentally changes this model.

Today:

- **Underwriters** can build risk analysis workflows.
- **Claims teams** can automate document reviews.
- **Operations teams** can automate reporting.
- **Product managers** can prototype applications.
- **Analysts** can create AI-powered assistants.

The rise of AI builders is creating a new category of employee: **the Business Builder**. Business Builders possess deep domain expertise, workflow knowledge, and AI tooling familiarity. They can rapidly prototype and validate ideas without requiring large development teams. The most successful organizations are finding ways to enable these builders rather than restrict them.

5 · The Emerging Risk: Shadow AI

As business users gain access to powerful AI tools, organizations face new risks. The solution is not to prohibit AI. The solution is **governed enablement**.

INDUSTRY DATA

Industry surveys consistently report that the **majority of knowledge workers use at least one AI tool their employer has not formally sanctioned**, and that most enterprise security programs have **limited telemetry** on how those tools handle company data.

Stanford AI Index · MIT-Sloan Generative AI in the Enterprise · Bain Generative AI surveys.

Security

- Unauthorized data access
- Sensitive information exposure
- Prompt leakage

Compliance

- Regulatory requirements
- Auditability
- Data residency

Operational Risk

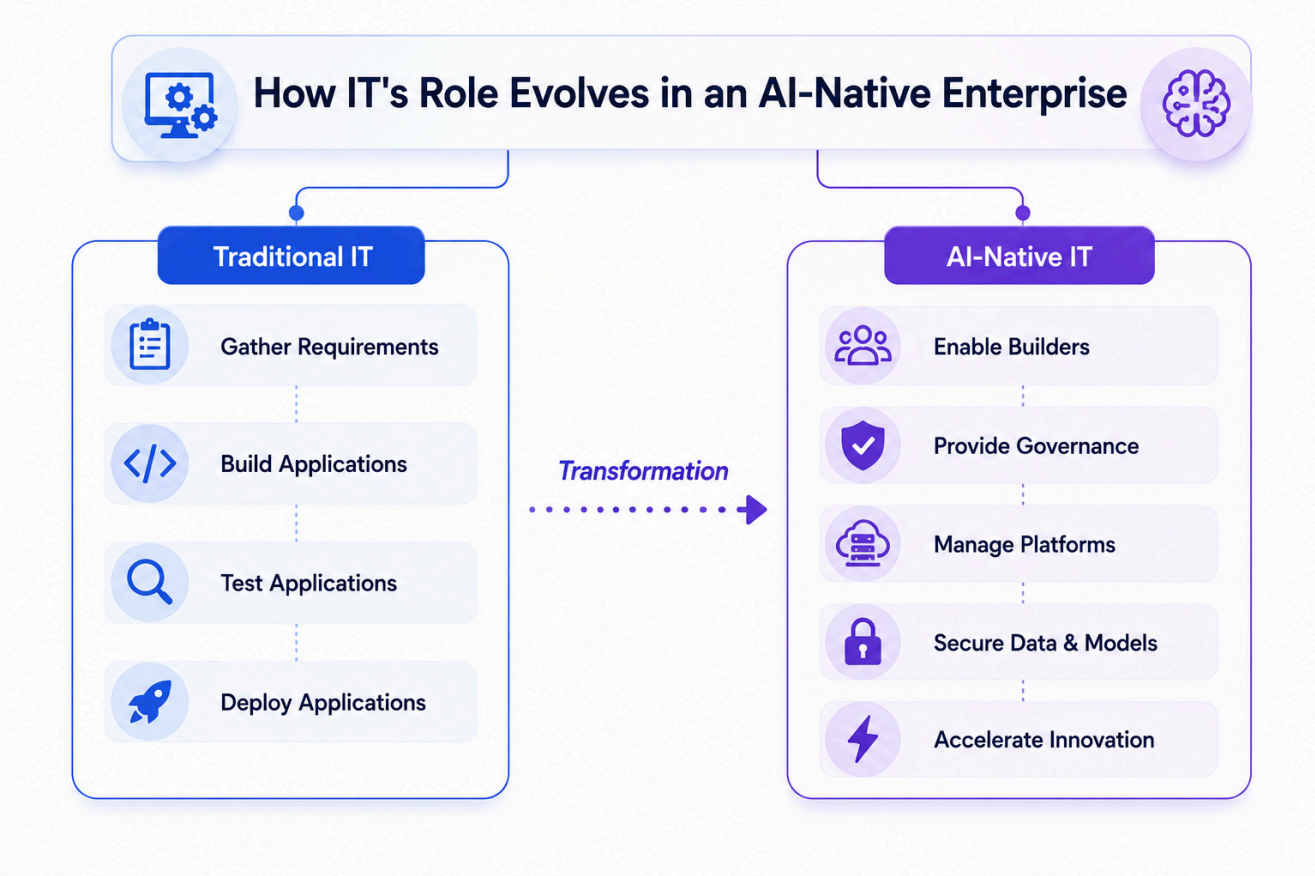
- Unsupported applications
- Lack of ownership
- Version sprawl

Model Risk

- Hallucinations
- Inconsistent outputs
- Bias

6 · The New Role of IT

Many leaders ask: "*If business users can build applications, what is the future role of IT?*" The answer is that IT becomes **more important, not less**. The future IT organization becomes an **innovation platform** rather than a software factory — focused on enabling builders, providing governance, managing platforms, securing data and models, and accelerating innovation.

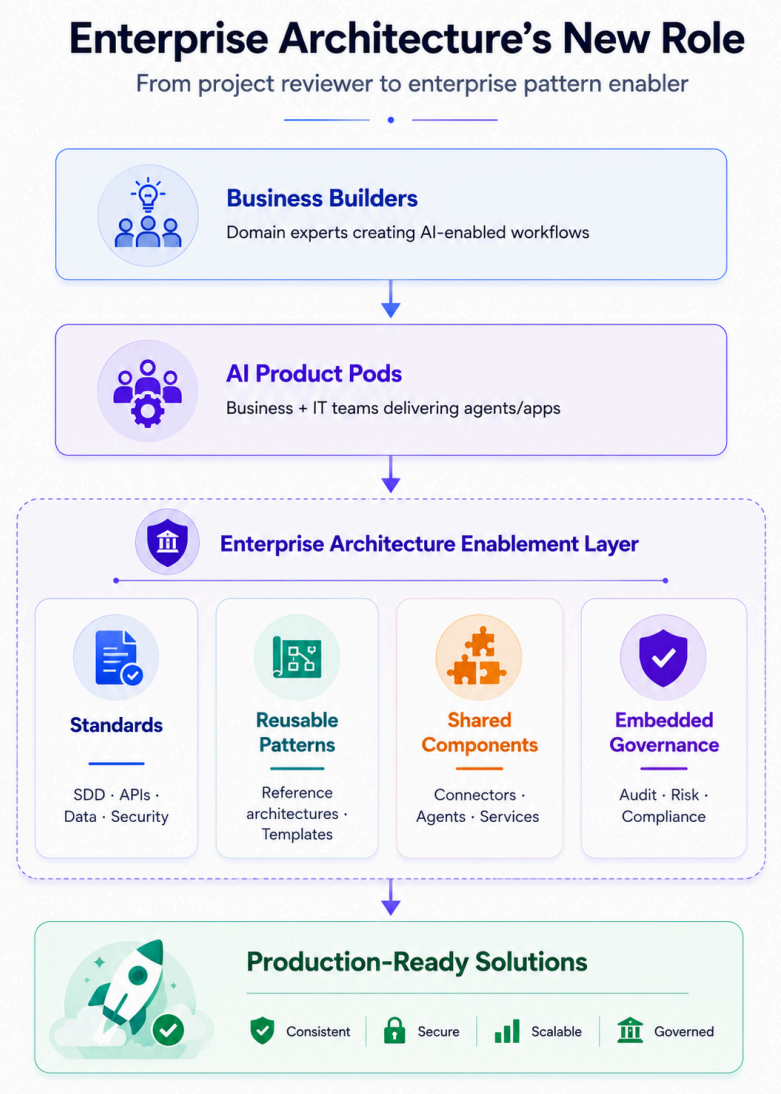


Traditional IT focused on building and testing applications. AI-Native IT shifts upstream — to enabling builders, providing the governance fabric, and accelerating organization-wide innovation.

7 · Enterprise Architecture's New Role

If IT's role moves from delivery to enablement, the role of Enterprise Architecture changes even more fundamentally. In the traditional model, EA reviewed projects — approving designs, raising concerns, gating releases. That posture worked when delivery cycles were measured in quarters. It cannot keep pace when Product Pods are shipping in weeks.

In the AI-native operating model, **Enterprise Architecture no longer reviews every project — it defines the standards, patterns, and reusable building blocks that every AI Product Pod inherits**. The EA team shifts from project reviewer to *enterprise pattern enabler*. Pods compose against EA's library; governance is applied by construction rather than by review.

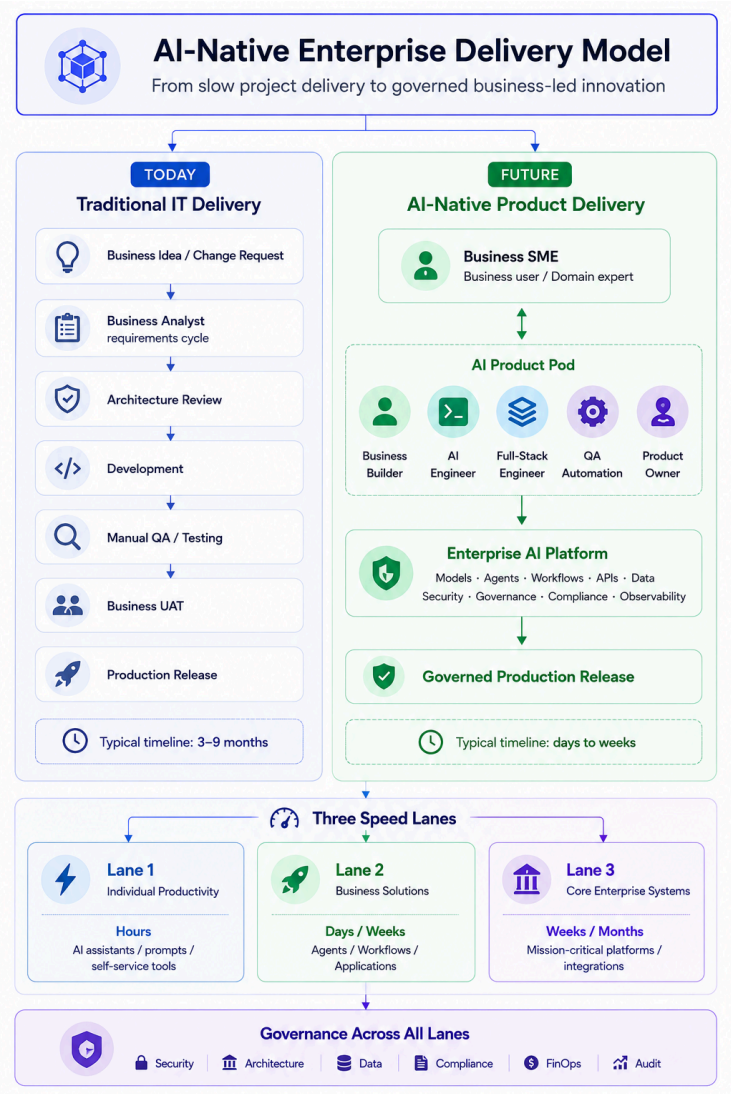


EA's contribution becomes the enablement layer — Standards, Reusable Patterns, Shared Components, Embedded Governance — that every AI Product Pod composes against. Pods get speed; the enterprise gets consistency, security, and reuse.

This shift is significant for two reasons. First, it makes EA structurally faster than the delivery work it once reviewed — building the patterns ahead of time rather than chasing the deltas afterwards. Second, it converts EA's intellectual capital into **reusable enterprise assets** — reference architectures, templates, connector libraries, governance controls — that compound in value with every pod that uses them. EA is no longer a gate; it is the platform on which the rest of the organization composes.

8 · The AI-Native Delivery Model · Three Speed Lanes & AI Product Pods

Leading regulated enterprises are converging on a new operating model: **three speed lanes** of delivery — Individual Productivity, Business Solutions, and Core Enterprise Systems — each with appropriate timelines, supported by cross-functional **AI Product Pods**, and unified by governance applied across all lanes. The contrast with the traditional 3–9 month delivery cycle is visualized below.



9 · Insurance Industry Examples

Four representative workflows where AI-native delivery is already collapsing duration from days or hours to minutes.

Underwriting Clearance

TRADITIONAL PROCESS

- OFAC checks
- Sanctions checks
- Capacity review
- Multiple systems
- Email coordination

Duration · **1–3 days**

AI-NATIVE PROCESS

- Parallel execution
- Automated recommendations
- Risk summary generation

Duration · **Minutes**

Submission Intake

TRADITIONAL PROCESS

- Broker emails
- PDF applications
- ACORD forms
- Manual triage

Duration · **Hours**

AI-NATIVE PROCESS

- Document extraction
- Data normalization
- Automated routing

Duration · **Minutes**

Statement of Values Processing

TRADITIONAL PROCESS

- Spreadsheet cleanup
- Data mapping
- Validation

Duration · **Several hours**

AI-NATIVE PROCESS

- AI extraction
- Validation
- Standardization

Duration · **Minutes**

Claims Investigation

TRADITIONAL PROCESS

- Manual research
- Multiple data sources
- Narrative generation

Duration · **Hours / days**

AI-NATIVE PROCESS

- Automated intelligence gathering
- Internal & external insights
- AI-generated summaries

Duration · **Minutes**

10 · AI Organization Maturity Journey

From isolated experimentation to enterprise-wide AI-native delivery — the journey unfolds across five levels. **Most regulated enterprises today sit between Level 2 and Level 3.** The organizations generating outsized value are rapidly moving toward **Level 4 — AI Product Pods**.

> **Where are you today?** Industry surveys consistently show widespread adoption of unsanctioned AI tools across knowledge workers, often outpacing formal governance programs. This is a clear signal that organizations are operating in Level 2 / 3 territory with the operating model lagging behind employee behavior. The transition to Level 4 is now *strategy work*, not *experimentation work*.

11 · The CIO Decision Matrix

The transition from a Traditional Enterprise operating model to an AI-Native Enterprise operating model touches every dimension of how IT and the business work together. The table below summarises the shift across the decisions a CIO is being asked to make over the next twelve to eighteen months.

DECISION AREA	TRADITIONAL ENTERPRISE	AI-NATIVE ENTERPRISE
Delivery	Projects	**Products**
Teams	Functional silos	**Product Pods**
Requirements	Documents	**Working prototypes**
QA	Manual	**AI-assisted**
Support	Ticket-driven	**Agent-assisted**
Innovation	IT-led	**Business-led**
Governance	Review gates	**Embedded controls**

None of these shifts can be made in isolation. Each one influences the others. Moving from project delivery to product delivery requires moving from functional silos to Product Pods. Moving from documents to working prototypes requires moving from manual QA to AI-assisted QA. **The CIO's job is to sequence these shifts deliberately rather than let them happen unevenly across the organization.**

12 · Example AI Product Pod · Submission Intake

An AI Product Pod is the smallest cross-functional unit capable of taking a business problem from idea to governed production. The canonical pod is **five roles** (as shown in §8): a Business Builder, an AI Engineer, a Full-Stack Engineer, QA Automation, and a Product Owner — partnered with a Business Sponsor from the line of business who supplies domain judgement and validates outcomes. The pod below — drawn from a real underwriting transformation — illustrates the typical shape: small team, weeks-long delivery, multiple agent outcomes, and benefits measured in business terms rather than throughput.

INSURANCE · UNDERWRITING

Submission Intake Pod

BUSINESS SPONSOR — **Underwriter (line-of-business SME)** — *supplies domain judgement, validates outcomes, owns the business case*

BUILDER	ENGINEER	ENGINEER	QUALITY	OWNER
Business Builder	**AI Engineer**	**Full-Stack Engineer**	**QA Automation**	**Product Owner**

Outcomes — Agents Delivered

- **Submission Intake Agent**
- **Clearance Agent**
- **SOV Agent**
- **Due Diligence Agent**

DELIVERY

4 weeks

BUSINESS BENEFITS

- Faster intake — minutes instead of hours
- Improved risk assessment — broader signal coverage per submission
- Reduced manual effort — underwriters focus on judgment, not data assembly

The same shape — a Business Sponsor paired with the canonical five-role pod, delivering a focused set of agents in a few weeks — is now being applied across claims triage, broker management, policy comparison, regulatory reporting, and other long-tail workflows traditionally trapped in the IT backlog. Pods scale up or down by adjusting how many of each role are present — a complex regulatory pod may carry two QA Automation engineers; a small automation may collapse Builder and Engineer into a single multi-skilled member. The roles are the contract; headcount per role is the variable.

Similar patterns are emerging across claims, finance, operations, compliance, legal, and customer service functions — wherever a clearly defined business workflow combines domain judgment with information assembly that AI can now perform reliably.

13 · What Leaders Should Do Next

The transition to an AI-native operating model is a programme of work, not a procurement decision. The phased plan below is the pattern used by enterprises that have begun this transition deliberately and successfully.

HORIZON 1
First 90 Days

- Identify 3 business-led AI opportunities
- Create 1 AI Product Pod
- Define AI governance standards
- Select an AI builder platform
- Define success metrics

HORIZON 2
Next 180 Days

- Expand AI Product Pods to additional business units
- Introduce AI-assisted QA
- Introduce AI-assisted AMS / support
- Create reusable agent patterns

HORIZON 3
Next 12 Months

- Stand up the Enterprise AI Marketplace
- Launch the Business Builder Program
- Establish the AI Governance Center
- Adopt the AI-Native Operating Model as the default

Each horizon builds on the last. The 90-day work establishes the foundations — one pod, one platform, one set of governance standards. The 180-day work expands the model across functions — QA and AMS join the transformation. The 12-month work institutionalises it — what was a pilot becomes the default way the enterprise builds and operates software.

Final Thought

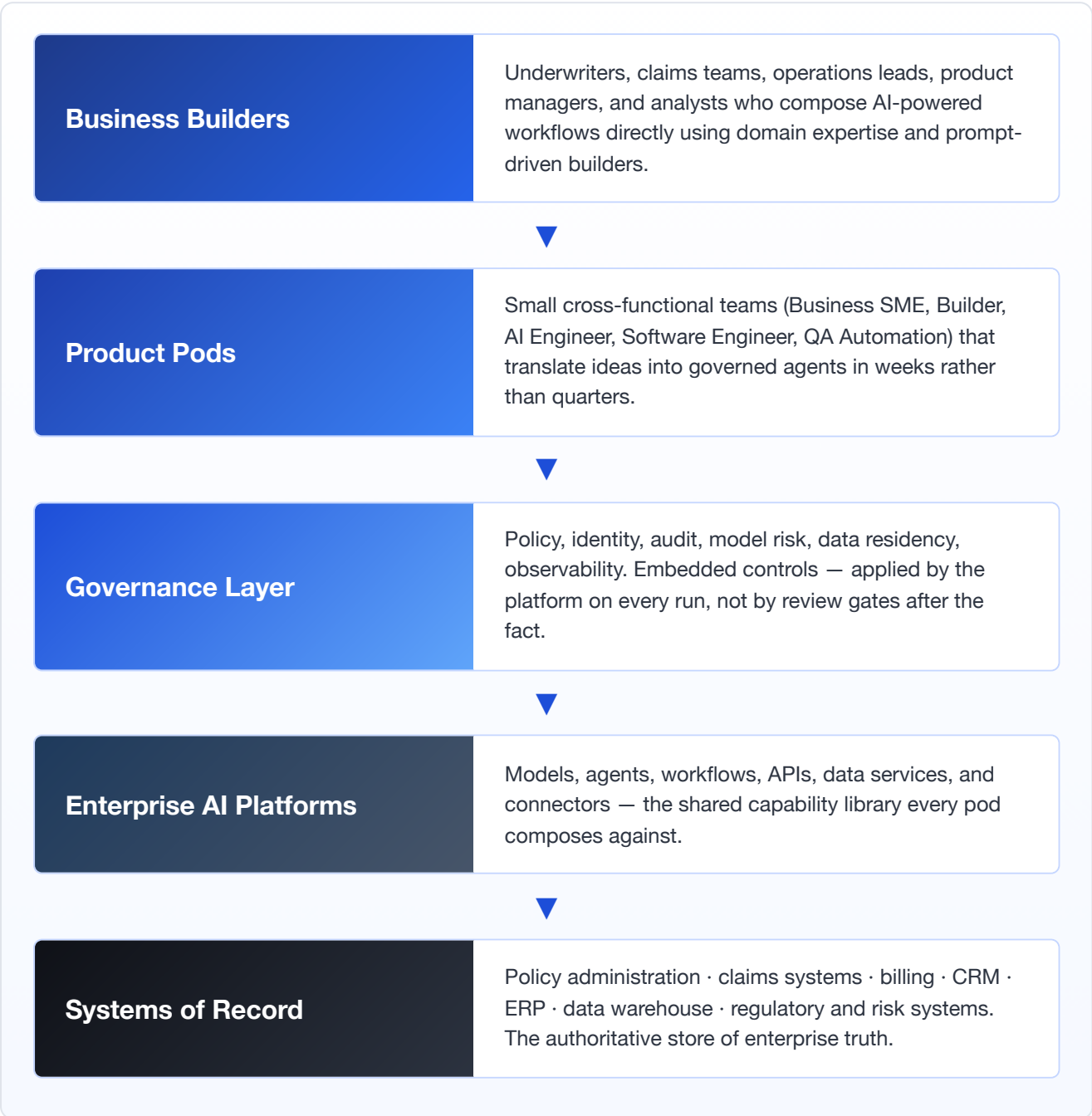
The next generation of competitive advantage in regulated industries will **not** come from access to AI models alone.

It will come from an organization's ability to combine **business expertise, AI-enabled builders, engineering excellence, and governance** into a new AI-native delivery model.

The organizations that master this transition will deliver innovation measured in **days rather than months** — while maintaining the trust, security, and compliance demanded by regulated industries.

14 · Reference Architecture for AI-Native Delivery

The shifts described in the preceding sections come together in a five-layer reference architecture. The top of the stack is where business outcomes are conceived; the bottom is where systems of record hold the enterprise's authoritative data. Every layer in between exists to translate the speed of the top into the stability of the bottom — without compromising either.



The architecture is layered deliberately. **Speed lives at the top.** Business builders and product pods iterate in days. **Stability lives at the bottom.** Systems of record change carefully and rarely. **The governance layer is the bridge** — it lets the top move quickly without putting the bottom at risk. The appendices that follow illustrate *representative implementations* of this architecture across business, development, QA and operations.

APPENDIX A.1 · BUSINESS TRANSFORMATION

From IT delivery queue to a governed business-app marketplace.

REPRESENTATIVE EXAMPLE · INSIGHTSTUDIO

Sales operations, customer success, finance, people ops, and product ops compose AI-powered internal apps directly on a governed platform. The long tail of departmental apps moves off the IT backlog; IT runs the fabric beneath every app.

TRADITIONAL

Business asks. IT builds. Eventually.

Departmental app requests queue on IT delivery for months; shadow IT (Power Automate, Zapier, browser-tab ChatGPT) fills the gap; customer data flows through unsanctioned tools.

AI-NATIVE

Business composes. IT enables.

Apps composed in a week, not a quarter; surfaced in an internal marketplace with Okta-style access; every app inherits SSO, audit, secrets, and connectors to Salesforce, SAP, Snowflake, SharePoint, Outlook.

ALL THE AVAILABLE AI TOOLS — AND WHERE INSIGHTSTUDIO FITS

ChatGPT / Claude · **Microsoft 365 Copilot** · **Salesforce Einstein** · **Power Platform + Copilot Studio**

ServiceNow App Engine · **Salesforce Agentforce** · **Lovable / v0 / Bolt** · **InsightStudio**

Chat assistants and copilots boost productivity inside existing tools. Prompt-to-app builders (Lovable, v0, Bolt) work for prototypes. Vendor platforms (ServiceNow, Salesforce, Microsoft) work inside those ecosystems. **InsightStudio is the pick when an enterprise wants a self-hosted, open, vendor-neutral fabric where business teams compose governed apps that integrate with every system — not just one vendor's.**

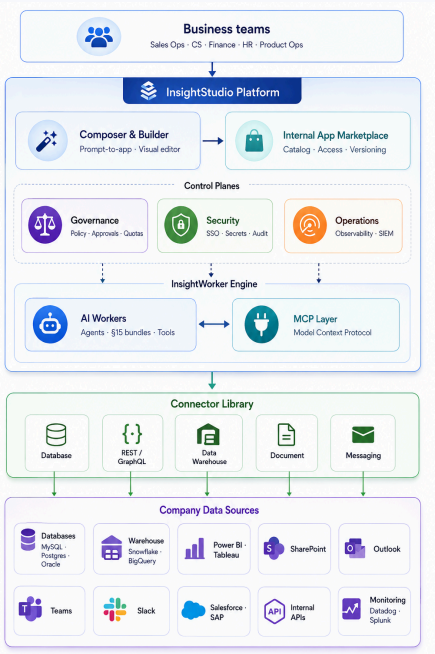
EXAMPLE IMPLEMENTATION

InsightStudio

AI application marketplace · No-code workflows · Business-led agents

- **Prompt-to-app composer** — describe the app; platform scaffolds workflow, form, outputs
- **Visual editor** — refine inputs / outputs / integrations without code
- **Governance & approvals** — admin review before any app surfaces in the marketplace
- **Enterprise controls** — SSO, secrets vault, audit log, observability on every run
- **Internal marketplace** — Okta-style discovery and access requests across teams

REFERENCE ARCHITECTURE



CUSTOMER PATTERN · TECHNOLOGY · MID-SIZE B2B SAAS

~1,800 employees, ~210 engineers on GitHub Copilot. Sales ops, customer success, finance, people ops, and product ops queued on IT delivery for departmental apps. Deployed InsightStudio; GitHub Copilot remained in engineering's IDE; InsightStudio absorbed the long tail of business apps that the engineering backlog could not reach.

APPENDIX A.2 · DEVELOPER TRANSFORMATION

Write the spec once. Agents derive the code, the tests, and the docs.

REPRESENTATIVE EXAMPLE · INSIGHTWORKER · DEVELOPER EDITION

Engineering organizations adopting spec-driven development with InsightWorker stop maintaining three artifacts in parallel. The spec becomes the source of truth; when it changes, the agent re-derives the implementation, tests, and documentation — and opens a pull request with the diff.

TRADITIONAL

Three artifacts drift apart over time.

Spec in Confluence; code in the repo; tests somewhere else. Engineer reads spec, writes code + tests + docs by hand. Spec changes; code is updated; tests and docs fall behind. Six months later, three artifacts tell three different stories.

AI-NATIVE

One spec. Three derived artifacts. Always in sync.

Spec authored once in a structured format. Agents derive the implementation, the test cases, and the API documentation. Spec change triggers re-derivation; the new diff lands as a pull request. The repo always reflects the current spec.

ALL THE AVAILABLE AI TOOLS — AND WHERE INSIGHTWORKER SDD FITS

GitHub Copilot (IDE) · Cursor · Tabnine · Claude Code · Devin · Aider · Lovable / v0 / Bolt

InsightWorker SDD

Keep Copilot in the IDE for code completion. Keep Claude Code and Devin for repo-wide refactors. Keep Lovable / v0 / Bolt for prototype exploration. **InsightWorker SDD is the layer that ties them together — making the specification the authoritative artifact and keeping implementation, tests, and documentation honest to it. It coexists with, rather than replaces, your existing developer-AI tooling.**

EXAMPLE IMPLEMENTATION

InsightWorker (Developer Edition)

Spec-driven dev · Multi-agent delivery workflows

- **Spec-driven development** — structured spec format covering inputs, behavior, success criteria, controls
- **Code generation** — language-native implementation derived from the spec, ready for engineer review
- **Test generation** — unit, integration, and contract tests derived in lockstep with the implementation
- **Architecture validation** — agent checks the implementation against the spec's success criteria
- **Documentation generation** — OpenAPI specs, Markdown docs regenerated on every spec change
- **Multi-agent workflows** — orchestrated agents pass artifacts through build → test → review → merge

REFERENCE ARCHITECTURE

CUSTOMER PATTERN · RETAIL · NATIONAL ENTERPRISE

Large engineering organization shipping new endpoints and back-office workflows continuously. Specs traditionally drifted from code within weeks; tests lagged; API docs were always behind a release. InsightWorker SDD now scaffolds new API endpoints from specs, generates ETL workflows for back-office processing, and regenerates API documentation on every spec edit.

APPENDIX A.3 · QA TRANSFORMATION

Product owners and test leads drive automated testing directly.

REPRESENTATIVE EXAMPLE · INSIGHTTESTBENCH

Paste a brief; the bench logs in via SSO, crawls every page, builds a feature tree, and generates regression / responsive / UX-audit plans grounded in real DOM. Failures diagnose themselves; confirmed bugs become pull requests. **When a structured spec from Appendix A.2 exists, the bench reads the same artifact — behaviors become test cases, success criteria become assertions — so code and tests stay aligned by construction.**

TRADITIONAL

Regression is a fortnight of QA effort.

QA engineers writing Playwright by hand; ~30% of CI failures dismissed as "probably the test"; UI changes break the suite; the skip-list grows release after release.

AI-NATIVE

Regression is a scheduled overnight job.

Product owner pastes a brief; the bench generates ~70-100 cases grounded in real DOM; nightly run with baseline compare; AI RCA on every failure with confidence + suggested fix; Slack pings only on real regressions.

ALL THE AVAILABLE AI TOOLS — AND WHERE INSIGHTTESTBENCH FITS

Manual Playwright · Cypress Studio · Selenium IDE · Mabl · Testim · Functionize · QA-Copilot in Cursor · **InsightTestBench**

Open-source test tools (Playwright, Cypress, Selenium) help engineers write tests faster. Vendor SaaS platforms (Mabl, Testim, Functionize) work but ship test data outside your environment. **InsightTestBench is the pick when an enterprise wants product owners and test leads to drive QA without writing Playwright — self-hosted, audit-first, with the bug-fix-agent loop closing all the way to a pull request.**

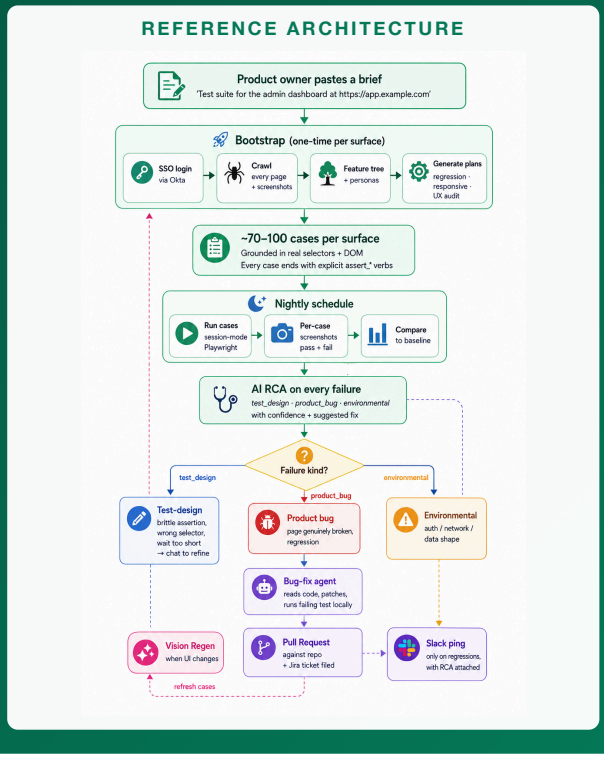
EXAMPLE IMPLEMENTATION

InsightTestBench

AI-generated cases · Regression suites · Bug-fix agent loop

- **AI-generated test cases** — bootstrap from a brief; grounded in real screenshots + DOM with explicit assert verbs
- **Regression suite generation** — per-feature plans, plus responsive (desktop / tablet / phone) and UX-audit suites
- **Test execution automation** — scheduled runs; session-mode Playwright; per-case screenshots on pass + fail
- **Release risk assessment** — every run compared to last green baseline; AI RCA on every failure
- **Bug-fix agent** — confirmed product bugs become pull requests against your repo, with Jira ticket auto-filed

REFERENCE ARCHITECTURE



CUSTOMER PATTERN · EMPLOYMENT MARKETPLACE · LARGE JOB PORTAL

Multiple customer-facing product surfaces; 28-person QA team writing Playwright. Regression cycle took two weeks per release; flake rate degrading trust. Bootstrapped 7 surfaces; product owners drive testing directly; QA team transitioned from Playwright authors to AI test operators.

APPENDIX A.4 · AMS / SUPPORT TRANSFORMATION

L1 support handled by agents. L2 engineers focus on the novel.

REPRESENTATIVE EXAMPLE · INSIGHTWORKER · OPERATIONS EDITION

AMS teams spend most hours on triage and routine tickets. InsightWorker agents take L1: triage every incoming ticket, resolve the routine ones autonomously, and hand the rest to L2 with logs, recent commits, and a proposed fix pre-attached.

TRADITIONAL

Engineers do the AMS investigation by hand.

L1 engineers triage incoming tickets manually; each ticket needs log search, recent commits, reproduce, diagnose; routine tickets eat 60-80% of AMS engineer hours; novel cases get the leftover attention.

AI-NATIVE

Agents do L1. Engineers handle the novel.

Every ticket triaged, classified, and routed in seconds. Routine tickets resolved autonomously with full audit trail. L2 receives tickets pre-prepared: log analysis, code changes, fix proposal. Engineer attention reallocated to novel cases.

ALL THE AVAILABLE AI TOOLS — AND WHERE INSIGHTWORKER L1 FITS

ServiceNow AI Agents · PagerDuty AI · Datadog AI · Atlassian Intelligence · Microsoft Copilot for Service · Custom RAG agents · **InsightWorker L1**

Vendor AI add-ons (ServiceNow, PagerDuty, Datadog, Atlassian, Microsoft) work well inside those ecosystems. Custom RAG agents work but each becomes its own maintenance project. **InsightWorker L1 is the pick when an enterprise wants self-hosted L1 / AMS automation that works across ticket systems, uses your own credentials, runs on your own infrastructure, and exposes every action in an open audit log — no vendor lock-in.**

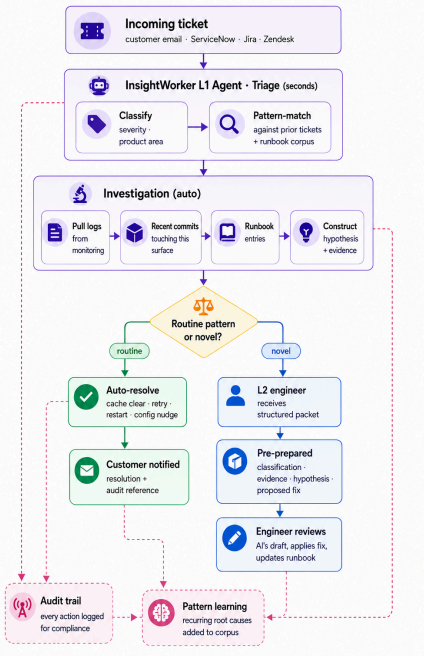
EXAMPLE IMPLEMENTATION

InsightWorker (Operations Edition)

L1 agents · Triage · Knowledge retrieval · Runbook execution

- **L1 support agents** — read incoming tickets, classify, route to L2 or auto-resolve
- **Incident triage** — pattern-match against prior tickets and runbook corpus
- **Knowledge retrieval** — pull logs, recent commits, runbook entries; construct hypothesis
- **Runbook execution** — routine cases resolved by the agent (cache clear, retry, restart, config nudge)
- **RCA assistance** — L2 receives a structured packet (evidence, hypothesis, proposed fix), not a raw ticket
- **Pattern learning** — recurring root causes added to the corpus, expanding auto-resolve over time

REFERENCE ARCHITECTURE



CUSTOMER PATTERN · BLOCKCHAIN · WORLD-LEADING FIRM

Global blockchain platform with a large AMS organization. Engineers consumed by triage and log analysis on routine tickets; backlog of novel cases lengthening; SLAs under pressure. InsightWorker now triages every ticket within seconds; routine resolved without engineer time; engineer attention reallocated to genuinely novel cases.