The Era of Application-Aware AI

Stephane H. Maes[1]

February 11, 2026[2]

Abstract

Escaping Pilot Purgatory with Real-Time Discovery & Coding (RTDC). Enterprise Intelligence, Instantly!

Despite an estimated annual capital allocation of thirty to forty billion dollars toward Generative Artificial Intelligence (GenAI), enterprise adoption remains severely constrained by the Deployment Paradox. Current industry data indicates that ninety-five percent of enterprise pilot projects fail to graduate to production environments. This failure rate is fundamentally a failure of integration architecture rather than an inherent limitation of language models. Early enterprise deployments have relied on attaching generic conversational agents to the periphery of legacy software ecosystems. This model-level integration approach introduces substantial friction, lacks contextual awareness, and forces engineering teams into the Stitching Trap, i.e., the manual construction of highly brittle application programming interface wrappers across poorly documented legacy environments.

This paper introduces the concept of Application-Aware AI, a novel architectural paradigm. Driven by a framework defined as Real-Time Discovery and Coding (RTDC), this approach operates as an autonomous entity that proactively discovers system logic, infers database schemas, and self-codes, under constraints, functional integrations dynamically based on user intent. The system executes a continuous four-layer loop encompassing total enterprise introspection, deterministic constraint enforcement, autonomous meta-agent orchestration, and dynamic user interface generation.

By abstracting probabilistic language models behind a strict Model of Constraints, and transforms, i.e., ~skills, and logging all decisions within a highly transparent Reasoning Graph, the proposed paradigm resolves the liability of model hallucination. This design ensures complete regulatory auditability, facilitates the progressive modernization of legacy enterprise applications, like ERP and ITSM, via the Strangler Fig pattern, and allows organizations to establish a production-ready intelligence factory instantly.

---

1. Introduction: The Rise of a New AI Category

[1] shmaes.physics@gmail.com

[2] *This is a public version of the original Zenera white paper. Some of the secret sauce and confidential information has been removed. More details can be found in other references [3,4,7-12,16-18], and in particular full technical and implementation details are in [7], and derived pending applications. This version has been published on March 27, 2026. The original is available with the right credentials on the Zenera web site [3]. Material, and references (in the text) introduced after February 11, 2026, are in italic.*

Cite as: Stephane H. Maes, (2026), "The Era of Application-Aware AI", https://zenodo.org/doi/10.5281/zenodo.19322700, https://shmaes.wordpress.com/2026/03/28/the-era-of-application-aware-ai/, February 11, 2026. (https://osf.io/vz6jt/files/e7pxd)

By early 2026, the enterprise AI technology market reached a critical inflection point defined by the "Deployment Paradox." Despite an estimated $30–40 billion in annual capital allocation toward Generative AI, a staggering 95% of enterprise AI pilots fail to graduate to production or deliver measurable financial returns [1].

Requirement	Why It's Critical	Copilot/RAG Gap
Domain Specificity	Generic models fail on specialized jargon (e.g., "PCIe lane bifurcation' or "GAAP revenue recognition').	X Copilots are generic. Fine-tuning is limited or expensive.
Tribal Knowledge	80% of process knowledge is heuristic ("If error X, check cable Y") and undocumented.	X RAG only finds documents. It cannot observe and learn from behavior.
Dynamic Data (RTDC)	Decisions must be based on millisecond state, and enterprise data and apps Requires real-time understanding of APIs and schemas.	X RAG relies on stale indices. Connectors hit rate limits. Integrations to Backends and apps are to be done, which is slow and hard
Agency & Workflow	AI needs to do the work (Plan → Code → Deploy), not just suggest it.	X Copilot is passive. Cannot execute multi-step loops or self-correct.
Guardrails & Audit	Regulated industries need a "Black Box" recorder for every AI action.	X Chat logs are insufficient. No code-level audit trail.

Figure 1: What the Enterprise Actually Needs (And Why Copilot-like approaches Fails). This is why many such enterprise stall, at least when in an heterogeneous, i.e., multi-vendor environment for data bases and enterprise applications.

This failure is not a result of insufficient intelligence in the models; rather, it is a failure of architecture. The first wave of Enterprise AI relied on "Model-Level" integration, bolting generic chatbots (e.g., Copilots or Conversational agents) onto the side of legacy applications (See figure 1). These tools lack context, cannot navigate the "Brownfield" reality of complex enterprise data, and introduce friction by forcing users to switch focus away from their core work. See figures 1, 2 and 3.

This paper introduces a new category of intelligence: Application-Aware AI. Unlike passive chatbots, Application-Aware AI is embedded directly into the application stack. Powered by Real-Time Discovery & Coding (RTDC), it acts as a (team) of Digital Worker(s), which autonomously, discovers system logic, schemas and APIs, self-codes integrations, and logic, and generates user interfaces (for reports, agents, applications, etc.) on the fly, based on a user intent. This architecture enables the Private AI Factory, i.e., a platform or OS that that industrializes intelligence to deliver the security, auditability, and operational scale required by regulated industries. With this platform Enterprise can instantly get intelligence.

2. The Deployment Paradox and the "Stitching Trap"

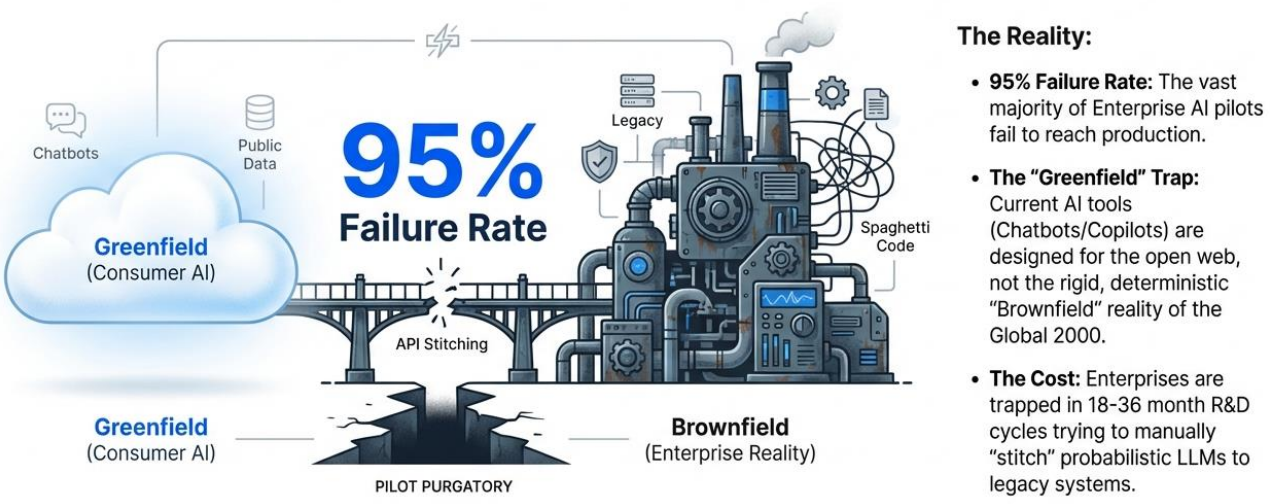

Figure 2: The Pilot purgatory and Stitching Trap, which is one of the main reasons why 95% of the enterprise AI projects fail: the people who bring AI into an enterprise do not know the legacy system and context of the enterprise. It takes a lot of time and efforts to do so.

The defining metric of the current market is the "GenAI Divide", i.e., the chasm between the 5% of organizations achieving ROI and the 95% stuck in "Pilot Purgatory". The primary driver of this divide is the "Stitching Trap". It is hinted in figure 2, based on [1]. Even the latest agentic AI initiatives encounter the same challenges [2].

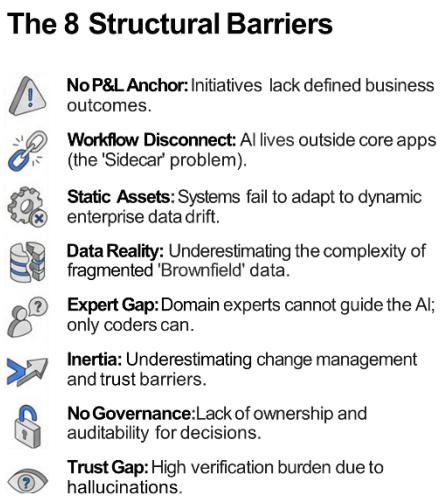

Figure 3: The complete list of reasons why enterprise AI project fail according to [1].

To connect modern AI to 20-year-old legacy systems, internal engineering teams typically attempt to "stitch" them together by manually building API wrappers, when APIs exists or are known, something not necessarily the case with for example old ERP or PSL SQL applications.

This approach is fundamentally flawed for a few reasons:

1. The API Bottleneck: Most legacy systems lack the clean RESTful APIs required by modern LLMs. Building these wrappers is a manual, brittle process that traps organizations in an 18-to-36-month R&D cycle. Also API may be missing, or little known.

2. Schemas: Data schemas and disparate repositories may be involved and again poorly known.
3. Maintenance Debt: "Static Agents" built via stitching break whenever a business rule or schema changes, whenever personas have new needs, or possibly when the backend applications are upgraded or migrated, e.g., if some APIs or customizations change. Industry data suggests maintenance consumes 35% of the initial investment annually, forcing IT teams to become infrastructure mechanics rather than innovators.

Yes, all these issues can be done but with costly efforts that take times. In this AI world, Speed is critical. Speed is life. And enterprises need their Enterprise AI instantly to compete or leapfrog the competition and reap the benefits of AI.

Application-Aware AI bypasses this trap entirely. Instead of asking developers to rewrite legacy systems to fit the AI, or to explain what schemas mean and what API t use, the AI uses RTDC to read, understand, and adapt to the legacy system in real-time [*3,4,7-12,17*].

3. Real-Time Discovery & Coding (RTDC): The Engine of Autonomy and Enterprise Application-Aware AI

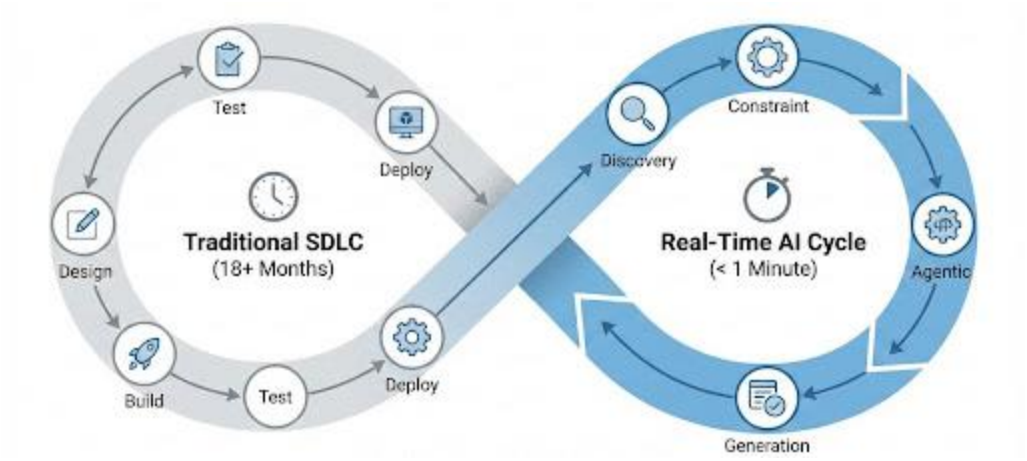

Figure 4: The run time evolution from traditional Software development life cycle to RTDC à la Zenera [3,4,7-12,17].

RTDC for Application-aware AI represents a new architectural pattern. It shifts AI from Design-Time assistance (helping developers write code) of AI and Vibe coding to Runtime execution (the AI *is* the developer right now, at runtime.). This process operates through a continuous, four-layer loop:

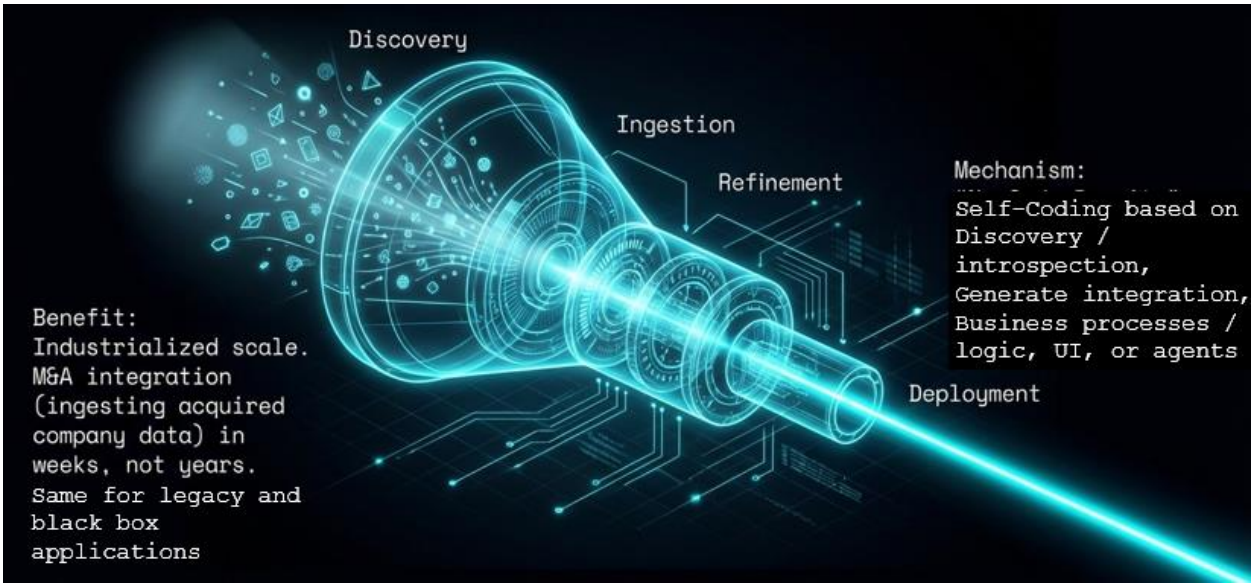

Figure 5: The AI factory based on RTDC [4,7,17]. All information about the enterprise context, including API, repositories, schemas, and documentation, manual, processes are ingested and / or discovered, and self-coding generates new logic and AI. Then experts verify and coach the AI to improve when needed.

3.1 The Discovery Layer (Total Introspection)

Consider figures 5 and 6. Before an AI can act, it must understand the Enterprise context. The Discovery Layer performs "Total Introspection" of the enterprise environment.

- Ingestion: It continuously scans the network to identify resources, not just documented "Golden Path" APIs, but also "Shadow APIs" and direct database connections. Scan can also e and processes can take place. Tribal knowledge can be similarly discovered.
- Schema Inference: It connects to SQL, NoSQL, and legacy mainframes to infer underlying data models, effectively reverse-engineering the system's logic without requiring manual documentation.
- Contextual Understanding: It ingests unstructured policy documents and compliance manuals, linking technical fields (e.g., "DOB") with semantic rules (e.g., "GDPR retention applies").

3.2 The Constraint Layer (The Guardian of Integrity)

To solve the "Black Box" liability problem of AI, where 5-15% of standard AI outputs are hallucinations, RTDC derives and imposes a deterministic **Model of Constraints**.

Figure 6: Real-time discovery, The brakes, and the Meta Agent for Enterprise Application- aware AI

Layers 2 involves:

- Laws of Physics: This layer encodes hard business rules (e.g., "Claims over $50k require senior approval") that the AI *cannot* violate.
- Audit & Explainability: Every action is logged in a Reasoning Graph, providing a "Glass Box" view of exactly why a decision was made and what data was accessed. This ensures 100% auditability for regulators like the NAIC.
- AI coaching: Experts can coach, i.e., correct or refine, the system the AI by examining the reasoning graph and output.

This way 100% correctness can be achieved (Figure 7), after subject matter expert spend enough time ensuring that the system well behave in a new context.

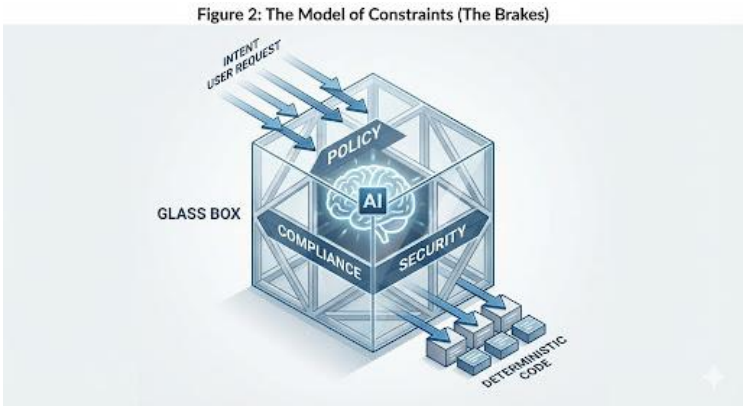

Figure 7: This approach ensure the possibility to achieve deterministic, 100% correct code and outcome as a result.

Defining primitives / skills, which we can transforms associated to what needs t be supported by targeted use cases in a target context allow us to be independent of the underlying LLM. For example switching from OpenAI ChatGPT to Google Gemini, was quasi immediate and provided the same performances. The same holds when going to an open source/open weight model to support on-premise or air gapped deployment, again does not result into performance degradation.

3.3 The Agentic Layer (Meta-Agents)

The core intelligence is driven by Meta-Agents—autonomous orchestrators that decompose high-level user intent into executable steps.

- Self-Coding: Unlike static bots with pre-canned responses, Meta-Agents write their own SQL queries, Python scripts, and API calls at runtime to fulfill requests. Note added on March 27, 2026: coding is the result of a conversation with the meta-agent. This is an example of AI coding or more exactly of vibe coding, which we know may have problems [*5,6,12,16*]. However, *as explained in [7,12],* the constraints and the execution of the whole Software Development Lifecycle (SDLC) by the same application aware platform ensures addressing most of the issues.
- Context Retention: They maintain the state of the conversation and user history, eliminating the "amnesia" that plagues generic LLMs. This is a transactional memory that also ensure that processed, and action are guaranteed to be executed (in Case of infrastructure failure), and that mistakes can be corrected.

3.4 The Generation Layer (Generative UI & Multi-Modal)

Application-Aware AI does not force users into a chat window. It uses Generative UI to construct the interface the user *needs* in the moment. In other words when we need to communicate with the user, an interface ca be dynamically created to do so, that it be to visualize results of to get input or approval. Once generated these ca now be mini-apps persisted for the future, for the user or for all users of a certain persona [*9,11*].

- Dynamic GUIs: If a user asks to "compare regional sales," the system generates a side-by-side bar chart and embeds it directly into the application dashboard.

Figure 8: Automatic and real time UI generation as logic is coded and added per a user intent. This can then persisted for the future, and/or certain personas. Note added on March 27, 2026: See for example [18], and figure 13.

- Multi-Modal Interaction: Users can interact via voice, text, or click. The system's "Visual-Voice UX" binds spoken intent directly to business logic, allowing field workers to execute complex ERP tasks hands-free (e.g., "Order a seal kit for Gearbox B").

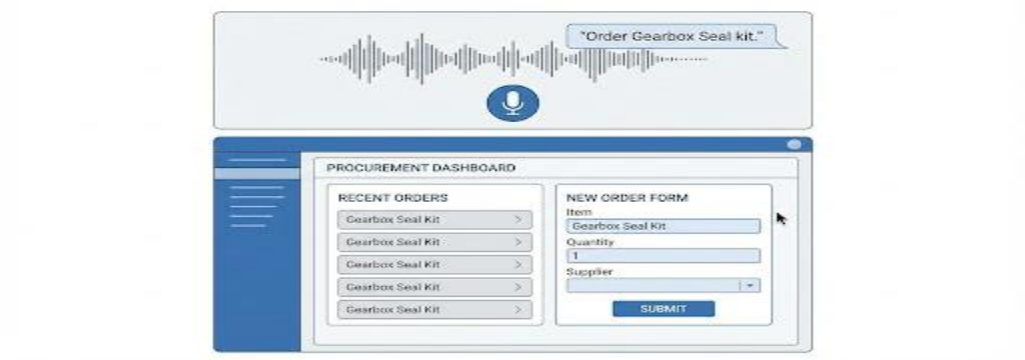

Figure 9: Example of auto generated multi-modal interface.

If desired, the generated new application can also be agent, i.e., agentic AI digital workers or chatbot like mini app with their corresponding interfaces. It can also be an integration.

The pattern itself can also generate new optimized business processes, or be used to instantly build (conventional) integrations.

4. The "Private AI Factory": Industrializing Intelligence

For the enterprise, the goal should be, not just "smart software" but an "AI Factory", an assembly line for instantly producing intelligent business outcomes. In fact, pilots and proof of concepts can now disappear: the first demo is a MVP version of a project. We really have enterprise AI instantly.

5. Coaching and Active Knowledge Transforms

A critical threat to the modern enterprise is the "Silver Tsunami"—the retirement of 50% of the insurance and utility workforce by 2028. These experts hold "Tribal Knowledge"—unwritten intuition about risk and systems.

Application-Aware AI captures this asset through Active Knowledge Transforms.

- Coaching the AI: Besides at discovery/injection, subject Matter Experts (SMEs) can "download" their expertise into the system using natural language. For example, a senior underwriter can explain, "I denied this claim because water damage occurred before the policy start date."

- Institutional Memory: The system converts this reasoning into deterministic rules within the Model of Constraints. This transforms the AI from a passive tool into an active repository of institutional wisdom, coaching junior staff with "In-Context Guidance" long after the experts have retired.

6. Application-aware AI

Summarizing all the features and capabilities discussed so far, we have introduced the notion of application-aware AI.

We define application-aware AI, a new way to bring intelligence, i.e., agentic AI, which works to enterprises now, where we have [4,7,8-12,17,18]:

- Resources, e.g., APIs, repositories, schemas, and meta-data, e.g., manuals, documentation, can be proactively discovered / scanned and ingested, (e.g., from an enterprise architect system), dynamically discovered to determine APIs to access them, their schemas, and build a model of constraints, e.g., limits[3] on what AI can do based on rules, policies, (best) practices, regulations, etc. It is what we denote as RTDC (Real-time Discovery and Coding)[4] [*4,7,8-12,17,18*].
- A meta-agent able, that can be configured via conversations, and can self-code new features and UI, possibly multi-modal, based on context/memory and conversations. All the actions of the agents can be sandboxed, explainable, coachable for refinement of AI and models, and traced at a transactional level, so that they can be long-lasting processes / applications. This way they can reach 100% accuracy and any issue can be rewind. Permissions can be linked to the original application/user RBAC.
- Persistent and sharable output as (across session or users) mini-applications, processes, UI and integrations, which provide outcome to an end-user interacted with them. Output is what does not exist yet. Integrations are dynamics as are new applications, AI agents, reports, dashboards and their UI per what is requested by the end user conversations.
- The underlying LLMs abstracted, achieved with transforms that are ~ skills, to allow smooth switch between LLM is a plus. It is consistent with prediction of commoditization of LLMs [13-16].

Zenera [3], has developed such an AI / agentic AI platform [*4,7,8-12,17,18*].

[3] Think of the laws of Physics for the enterprise.
[4] Note that others than Zenera may have followed and related road by network sniffing the actions of UI action on the repository to infer poorly known schema and APIs. However that approach will be limited in completing the agentic implementation of some inner workflow assumed as second step of the agentic strangler fig pattern discussed later, and in [*6,10*].

7. Strategic Use Cases: From Infrastructure to Intelligence

The pattern of RTDC for Enterprise Application-aware AI, is generic and horizontal across industries. The transition to Application-Aware AI is already solving high-value pains across many industries,

Zenera has also shown with customers that it is very well suited and proven in several industries, with the expectation that this generalizes to many others) [3].

- Global Insurance & Risk: A major carrier utilized the platform to automate high-speed claims processing. By ingesting unstructured police reports and validating them against policy constraints, the system automated data entry into legacy mainframes, moving humans from "data entry" to "decision review" and reducing processing time by 50%. This can also be used in BPO (Business Process Outsourcing) use cases.
- Infrastructure Operations: A global technology leader deployed "Intelligent Assist" to manage complex software stacks . The AI connects to specialized APIs to provide real-time diagnostics and automated remediation, reducing the "toggle tax" of switching between management tools.
- Logistics & Manufacturing: SMEs can now generate "Impossible ERP" customizations. A logistics provider tracking temperature-sensitive cargo can use RTDC to auto-discover IoT and maritime traffic APIs, generating a custom dashboard to correlate "frozen tuna temperature" with "port delays", i.e., a capability that would previously require millions in custom development. The vibe coding part can now be used to say do vibe design of pieced with say CAD output. We have been able to demonstrate that with minimum additions other than having to add the ability to render the CAD designs.
- Healthcare: The system can allow to do always expanding complex investigations / analytics across a wide set of disparate applications and repositories.
- ERP and Enterprise Applications: The system can implement agentic workers interacting autonomously with the ERP systems, 24 hours a day, 7 days a week, 365 days a year, and horizontally scalable at infinitum with almost no extra cost *(Note added on March 27, 2026: See [7,8,10,18])*. The processes can surround and add agentic capabilities first by implementing agentic worker processes, then as needed implementing internal processes[5]. This follows the Strangler fig pattern as shown in Figure 10. We predict that this will fundamentally disrupt the market for enterprise applications *(Note added on March 27, 2026: See [8,10]).*

In particular, the architecture of Zenera is LLM independent (i.e., Commercial or open source) and, as a result, can be deploy in the cloud, on-premises or even air-gapped, this way addressing many of the concerns that some regulated / security conscious industries have [3].

Eventually, the approach ensures that the system can ingest disparate data schemas from say a new acquisition, which would have different backend systems and repositories, in weeks rather than years, solving the integration failures that plague 50% of M&A deals. It illustrate why we can consider that such an approach is future proof.

Note text added on March 27, 2026:

Figure 10, reproduced from [8,10], sketches the strangler fig pattern built with application-aware agentic AI. Relying on RTDC and the other features of an application-aware AI platform, it is possible to progressively surround

[5] E.g., the manufacturing scheduling optimization agent shown in Figure 13.

existing / traditional enterprise applications like ERP or ITSM with agentic extensions, which often are agentic workers performing tasks like a human user, more efficiently with more data, 24hours a day, 7 days a week and 365 days a year. Incumbent enterprise application vendors are not really re-implementing more internal workflows (think of ITIL flows in ISM/ESM, or say production scheduling optimization in ERP), because they are paralyzed with the traditional challenges of the innovator's dilemma [SM]. This opens opportunities for new entrants[6]. With this approach, enterprises can also develop their own optimized agentic ERP, cheaper and optimized for their preferred business processes.

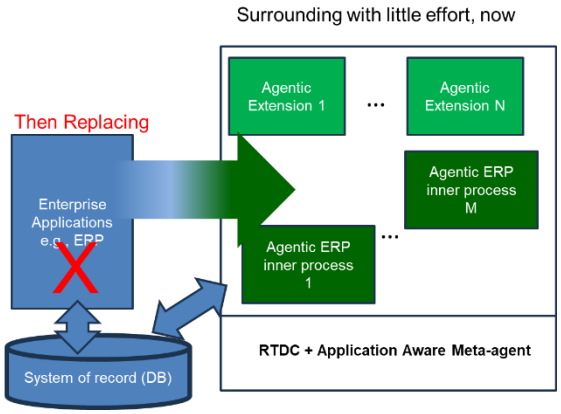

Figure 10. This figure, from [8,10], shows how agents built on an application-aware platform, can implement an agentic strangler fig pattern, first adding agentic capabilities to a traditional ERP, eventually reimplementing all the ERP processes that matters, allowing removal of the original ERP, at best just keeping the system of record DB.

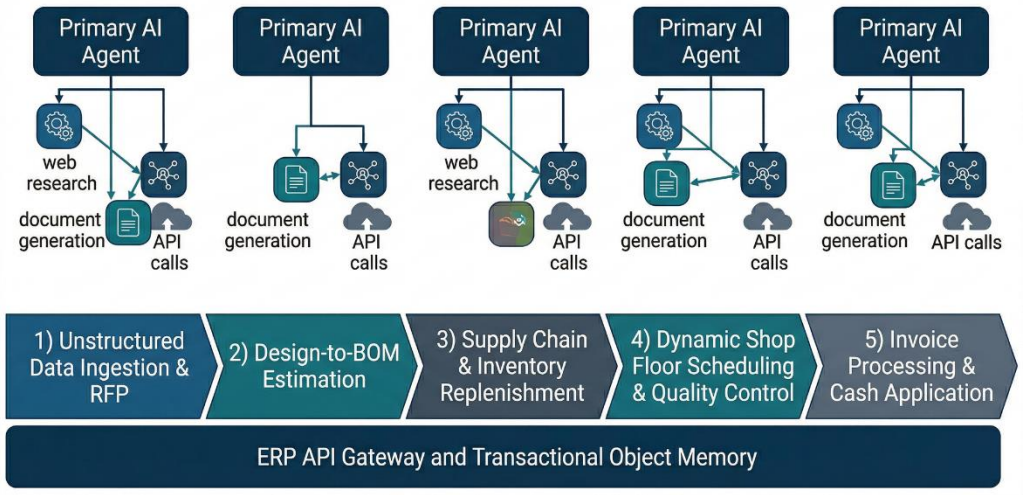

Figure 11: Autonomous Manufacturing Value Stream from Order to Cash and its agentic ERP workers (from [18]). Each can 1) help having more efficient end to end enterprise processes that transform the enterprise 2) have immediate ROI by making employees more efficient (and less employee required). An end to end agentic optimized value stream for a manufacturing company from order to shipping and cash, can be more efficient. The figure sketches a simplified example of workflow showing how the different agentic use cases / agentic workers contribute to realize the end to end value stream agentic, while also being individually reusable. These agentic workers work just like a human co-worker would: by using ERP and other enterprise systems and machines. They reason,

---

[6] *Note added on March 27, 2026: As discussed in [8,10], this does not mean the end of SaaS.*

delegate, search, design, build, monitor, handle events, remember, code, and deliver. This way, an end to end value stream is broken into tasks and subtasks, creating sub-agents for execution. The sub-agents might do web research, document generation, data processing, or API calls to your connected services. For example to answer an RFP, a document can drafted by one agent while another gathers the data it needs.

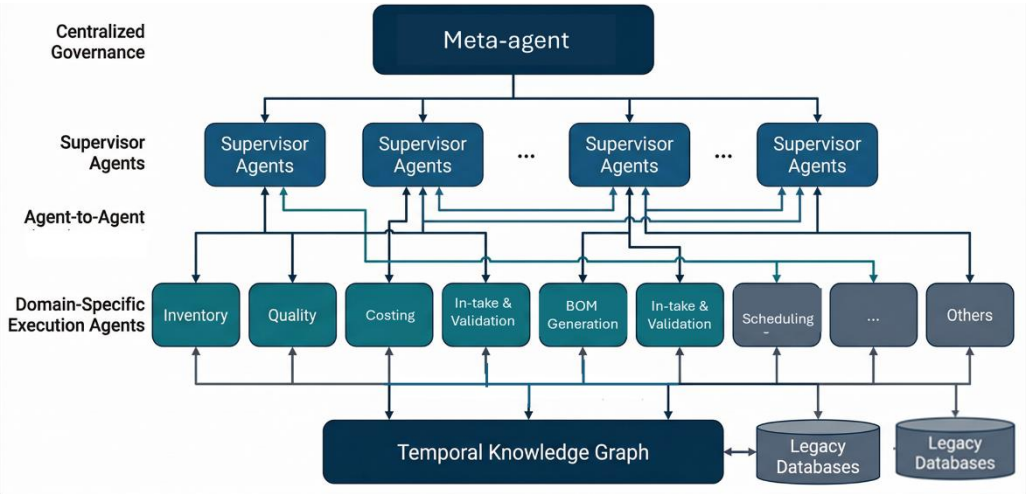

Figure 12: Agentic AI Agents Complement and make traditional Enterprise Applications Agentic (from [18]). This way they can also be customized to specific need of customers.

Figure 13, shows an example of dynamically built UI, used to communicate with the end user.

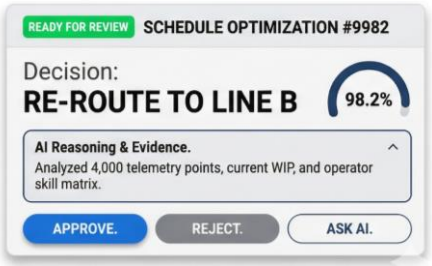

Figure 13: Dashboard for schedule optimization (from) [18].

The ROI of such a transformation to agentic AI ERP is significant due to rendering employees more efficient, which, for example, allows running the business with less resources, or better yet being able to increase manufacturing capacity with the same resources [*18*].

8. Conclusion: The Path to Enterprise Autonomy

The 95% failure rate of current AI projects is a signal that the Internal Build (or outsourcing, conventional 3rd party development shop or integrators) strategy is obsolete, unless relying on the RTDC Enterprise Application-aware AI pattern.

To cross the GenAI Divide, enterprises must abandon the stitching trap and adopt an architecture designed for the complexity of the real world.

Zenera Application-Aware AI with RTDC offers this path [*3,4,7-12,17,18*]. By embedding intelligence that can discover, constrain, and code itself, organizations can:

1. Escape Pilot Purgatory: Move from demo to production in days, not years.
2. Eliminate Technical Debt: Stop building brittle API wrappers.
3. Ensure Compliance: guarantee 100% auditability with the Model of Constraints.

The future belongs to software that evolves. It is time to turn your infrastructure into an Intelligence Factory.

With Zenera Application-Aware AI with RTDC, Enterprise AI can be instantly achieved. It is a key enabler to reach autonomous enterprises. It is a new category of AI. Indeed other AI approaches: LLM-based chatbots / copilot-like, agentic AI etc., can benefit of it to immediately offer value and ROI to enterprises.

With application-aware AI, using the agentic strangler fig patter, enterprise applications, like ERP and ITSM, can be evolved to agentic enterprise applications. We predict a disruption of these markets.

———

References

[1]: Aditya Challapally et al., (2025), "The GenAI Divide. State of AI in Business 2025.MIT NANDA, https://mlq.ai/media/quarterly_decks/v0.1_State_of_AI_in_Business_2025_Report.pdf, July 2025.

[2]: Gartner (2025). Predicts 40% of Agentic AI projects will be canceled by 2027 due to unclear value and risk controls.

[3]: Zenera, (2026), "The Enterprise AI Agent Factory. Your enterprise is unique. Your AI agents should be too. The Enterprise AI Problem, Solved", https://zenera.ai.

[4]: Stephane H. Maes, (2026), "Achieving Zero-Effort, Quasi-Zero Cost Integration with Zenera RTDC, and Application-Aware AI", Zenera, February 2026.

[5]: Stephane H. Maes, (2025), "The Gotchas of AI Coding and Vibe Coding. It's All About Support And Maintenance", https://doi.org/10.5281/zenodo.15343349, https://shmaes.wordpress.com/2025/04/28/the-gotchas-of-ai-coding-and-vibe-coding-its-all-about-support-and-maintenance/, April 28, 2025, (https://osf.io/kjz9t/download/).

[6]: Stephane H. Maes, (2025), "Ensuring the Maintainability and Supportability of "Vibe-Coded" Software Systems: A Framework for Bridging Intuition and Engineering Rigor", https://doi.org/10.5281/zenodo.15354102, https://shmaes.wordpress.com/2025/05/06/ensuring-the-maintainability-and-supportability-of-vibe-coded-software-systems-a-framework-for-bridging-intuition-and-engineering-rigor/, May 6, 2025, (https://osf.io/2nu8r/download/).

[7]: Stephane H. Maes, et al. (2026), US Patent Application, "AI System and Method of Meta-Agent and Application-Aware AI, Including Real-Time Discovery and Coding, for Deterministic Constraint Modeling, and Runtime Evolution of Software Applications", 2026.

[8]: Stephane H. Maes, (2026), "Agentic Smart ITIL, And The Disruption Of The Market Of Conventional Enterprise Applications", https://zenodo.org/doi/10.5281/zenodo.18870309, https://shmaes.wordpress.com/2026/03/04/agentic-smart-itil-and-the-disruption-of-the-market-of-conventional-enterprise-applications/, March 1, 2026. (https://osf.io/vz6jt/files/erw23).

[9]: Zenera, "Enterprise Analytics Reimagined, Build live reports and dashboards in minutes – no pipelines, no coding, https://zenera.ai/zeneraanalytics. Retrieved on March 5, 2026.

[10]: Stephane H. Maes, (2026), "Agentic AI, The Obsolescence of The Enterprise Application & Zenera, The Catalyst", Zenera, February 2026.

[11]: Zenera, (2026), "Zenera: From Conversation to Production in Minutes", YouTube, https://zenera.ai/introvideo. Retrieved on March 1, 2026.

[12]: Stephane H. Maes, (2026), "Vibe-Coding and SDLC Constrained And Managed By An Application-Aware AI-Like Agentic Platform", https://zenodo.org/doi/10.5281/zenodo.18972674, https://shmaes.wordpress.com/2026/03/11/vibe-coding-and-sdlc-constrained-and-managed-by-an-application-aware-ai-like-agentic-platform/, March 11, 2026. (https://osf.io/vz6jt/files/yv2xf).

[13]: Stephane H. Maes, (2025), "The Circle of Life for LLMs. Was the Reaction to DeepSeek Justified?", https://zenodo.org/doi/10.5281/zenodo.14838733, https://shmaes.wordpress.com/2025/02/15/the-circle-of-life-for-llms-was-the-reaction-to-deepseek-justified/, February 5, 2025. (osf.io/j2vsz_v1, vixra:2503.0009v1).

[14]: Stephane H. Maes, (2024), "Fixing Reference Hallucinations of LLMs", https://doi.org/10.5281/zenodo.14543939, https://shmaes.wordpress.com/2024/11/29/fixing-reference-hallucinations-of-llms/, November 29, 2024. (osf.io/u38w4/, viXra:2412.0149v1).

[15]: Stephane H. Maes, (2024), "The Trouble with GenAI: LLMs are still not any close to AGI. They will never be", https://zenodo.org/doi/10.5281/zenodo.14567206, https://shmaes.wordpress.com/2024/12/26/the-trouble-with-genai-llms-are-still-not-any-close-to-agi-they-will-never-be/, December 25, 2024. (osf.io/qdaxm/, viXra:2501.0015v1).

[16]: Stephane H. Maes, (2026), "Evaluating the Efficacy of Artificial Intelligence in Software Engineering: A Post-February 2026 Analysis", https://doi.org/10.5281/zenodo.19103818, https://shmaes.wordpress.com/2026/03/16/evaluating-the-efficacy-of-artificial-intelligence-in-software-engineering-a-post-february-2026-analysis/, March 13, 2026. (https://osf.io/vz6jt/files/gj4z2).

[17]: Ramu Sunkara, Stephane H. Maes, (2026), "Awareness Is the Real Differentiator in Enterprise AI", LinkedIn post, March 25, 2026.

[18]: Stephane H. Maes, (2026), "Transforming Manufacturing Operations with Agentic ERP", Zenera, March 27, 2026.