# Bridging the 94% Enterprise AI Value Gap: The Case for a Unified AI Operating System (AI OS)

## Connecting CRM, ERP, institutional memory, and governed execution across MSMEs and large enterprises

November 2026


## Abstract

Individual AI assistance can leave a business dependent on disconnected stock checks, credit approvals, and manual order entry. This paper examines how a unified AI operating system could connect those activities through shared records, institutional knowledge, and controlled execution. The analysis covers customer relationship management (CRM), enterprise resource planning (ERP), and commerce, with particular attention to the conditions under which an agent may make a business commitment. It considers implementation in Indian micro, small, and medium enterprises (MSMEs) and large enterprises, including data preparation, operating costs, and management responsibility. MatruOS supplies an applied architectural context. The method combines published research with analysis of an illustrative business workflow. The central proposition is that value should be evaluated across the complete workflow, including approvals, exceptions, recovery, and verified completion. The paper identifies requirements for subsequent empirical testing; its conclusions are architectural propositions whose financial effects require measurement in actual deployments.



Keywords: AI operating system, enterprise AI, CRM, ERP, institutional memory, agent execution, Indian MSMEs, operational governance.


## 1. Introduction and interpretation of the enterprise AI value gap

McKinsey’s August 2026 survey reports that 80 percent of respondents experienced individual productivity benefits, while 37 percent attributed some impact on enterprise earnings before interest and taxes (EBIT) to AI. Its high performer category, approximately 6 percent of respondents, combines an attributed EBIT contribution of at least 5 percent with a reported significant impact. The title’s 94 percent identifies respondents outside that category. It does not measure deployment failure or financial loss. The survey also reports agent scaling among 40 percent of respondents at organisations with annual revenue above one billion US dollars and 22 percent at smaller organisations. These figures describe different populations and measures, which must be interpreted separately. [1]

The research problem concerns how useful activity contributes to fulfilled commitments and measurable business outcomes. Additional drafting capacity has limited operational meaning if each resulting document requires extensive reconciliation before anyone can act on it. Time released from administration can nevertheless have practical value before an earnings contribution becomes identifiable. Evaluation should account for both the operational change and the conditions required for that change to affect financial results.

Brynjolfsson, Rock, and Syverson examine how complementary investment in intangible assets affects the measurement of productivity gains from general purpose technologies. Their analysis provides a reason to consider organisational investment and measurement delays alongside software architecture. Fragmentation

is therefore one possible contributor to the enterprise value gap, whose influence must be examined with other implementation conditions. [2]

This paper examines a specific architectural proposition: shared operational context and governed execution can reduce avoidable coordination work when a business process spans several functions. The proposition requires investigation because the same integration can introduce concentrated dependencies, migration costs, and security risks. A unified system must therefore be evaluated through the obligations it can fulfil reliably, the controls it preserves, and its total operating cost.

### 1.1 Research question and scope

The principal question is: under what conditions can an AI operating system convert access to business information into authorised, consistent, and verifiable execution across departments?

The scope is an enterprise management and execution environment covering CRM, ERP, commerce, and associated workflows. The term operating system refers here to the coordination of business information, authority, and action. Existing AIOS research applies operating system concepts to the management of large language model (LLM) agents. This paper examines the requirements that arise when those agents interact with business records and obligations. [3]

The method combines selected published sources with conceptual analysis of an illustrative workflow. Sources were selected for their relevance to enterprise AI adoption, complementary organisational investment, agent infrastructure, transaction integrity, and evaluation. This is a focused synthesis rather than a systematic literature review. The constructed examples identify architectural requirements and questions for subsequent testing.

## 2. Why individual task assistance may not produce complete business outcomes

### 2.1 Fragmented context and inconsistent commitments

Consider a distributor receiving a customer enquiry through WhatsApp. Customer history is held in the CRM, stock availability in an inventory application, outstanding balances in accounting software, and delivery arrangements in a separate operational record. A quotation agent may have enough information to produce a document but insufficient information to make a valid delivery commitment.

Suppose the requested goods appear available, but part of that quantity has already been reserved for another customer. The quoted price may also depend on an expired agreement, while a recently received payment has not yet been allocated to the customer ledger. Each source may be internally plausible. The problem arises when the agent combines them without knowing their effective time, authority, or relationship to the proposed transaction.

An integrated architecture should make those dependencies explicit. It must distinguish physical stock from available stock, account balances from authorised credit exposure, and published prices from agreements specific to the customer. It must also identify when a required fact is missing. In that circumstance, escalation is a valid outcome. Inventing a value or assuming that an earlier approval still applies creates an unsupported business commitment.

### 2.2 Coordination work between applications

The cost of a workflow includes the effort required to move from one valid state to the next. That effort can include copying data, resolving conflicting identifiers, checking approvals, and confirming whether another department completed its part. AI assistance may reduce work within one step while leaving these dependencies intact.

For example, an employee can prepare a quotation using AI and then manually check stock, request approval, enter the order, and notify dispatch. Measuring only quotation preparation omits the work required to make the transaction executable. A useful evaluation therefore follows the order through acceptance, reservation, fulfilment, invoicing, and reconciliation, while recording where human intervention remains necessary.

A specialised agent can be useful when its inputs, authority, and completion conditions are well defined. The architecture must identify which event constitutes success for that agent and which business obligations remain open. For an order workflow, preparing the quotation, reserving stock, and confirming dispatch are separate events with different evidence requirements.

### 2.3 Repeated context assembly and operating costs

Disconnected agents may repeatedly retrieve and interpret the same material, maintain inconsistent policy copies, or request duplicate confirmations. These behaviours create costs in model usage and employee attention. The relevant question is how much of that repetition is necessary for validation and how much arises from an avoidable lack of shared state.

A common context service can support reuse when it tracks the conditions under which information remains applicable. A cached credit decision requires review when exposure changes, while a pricing policy needs an effective date and an owner. The design must balance retrieval effort with the consequences of acting on outdated context.

## 3. Defining a unified AI operating system

Padhi and colleagues describe an AI operating system as a management layer that brings institutional knowledge into decision making and action. They report initial value in approximately twelve weeks in their deployment experience, alongside changes in planning cycle duration. These observations provide context for the proposed architecture. Their applicability to another deployment depends on its scope, starting conditions, and method of evaluation. [4]

For this paper, a unified AI operating system is an enterprise environment that maintains consistent business context, delegates bounded actions, records decisions and execution, and supports verification and recovery. Four connected layers organise the proposed architecture.

### 3.1 The transactional layer

The transactional layer holds authoritative records such as customers, products, stock movements, orders, invoices, and payments. Each record needs a defined owner and rules governing valid changes. A natural language request must eventually resolve to identified entities, validated values, and permitted state transitions.

A shared relational database can simplify some coordination requirements, particularly in a new deployment. An established enterprise may retain several systems of record. In that case, unification depends on consistent

identifiers, explicit ownership, integration contracts, and reconciliation. The choice between physical consolidation and integration should be assessed against the consistency requirements of the workflow and the cost of maintaining them.

### 3.2 The semantic context and institutional memory layer

Operational decisions depend on information that ordinary transaction tables do not fully express. Examples include approved discount conditions, reasons for supplier restrictions, customer commitments, and the circumstances under which management accepted an exception.

Institutional memory should record the source, authorisation, scope, effective date, and review conditions of business knowledge. A concession granted for one transaction needs to retain that scope, while historical explanations should remain distinguishable from current instructions. When instructions conflict, the system needs a defined precedence rule or a responsible person to resolve the conflict.

An agent should retrieve only the context needed for the task and permitted for its role. A customer communication workflow may need order status without access to unrelated employee information or another customer's commercial terms. Context availability and access control must be designed together.

### 3.3 The orchestration and execution layer

The execution layer converts an authorised intention into controlled operations. It identifies the action, checks preconditions, obtains any required approval, performs the permitted change, and records the result. The language model may assist with interpretation or planning; deterministic business services should enforce financial rules and record integrity.

Agents require permissions scoped to their assigned actions, with checks enforced when each action is attempted. A request to change a delivery address, for example, may require different authority before and after dispatch. Discount approval and credit limit changes also require separate authorisation because they affect different business obligations.

Database transactions can protect related internal changes, but they do not make an external message or payment service part of the same atomic operation. PostgreSQL's documentation explains that concurrent transactions may encounter serialisation failures requiring the entire transaction to be retried. Application logic must handle those outcomes explicitly. [5]

For a workflow that also sends an external confirmation, the system should durably record the intended communication and its delivery status. It should prevent an uncertain response from producing duplicate commitments. A timeout must be treated as an unresolved outcome until the external state has been checked.

### 3.4 The economics and governance layer

The governance layer defines who may act, under which conditions, and with what evidence. It also records the cost of reaching the outcome. These responsibilities belong in the workflow design because an action can be technically successful while exceeding a financial limit or consuming disproportionate resources.

Routine arithmetic, validation, and record updates should use deterministic functions where the rules are known. Model reasoning can be reserved for tasks involving interpretation, ambiguity, or synthesis. The choice of model should reflect task requirements, permitted data exposure, latency, and evaluation results.

A complete execution record should connect the request, relevant policy version, approval, attempted actions, final state, and unresolved exceptions. This allows an investigator to establish which information and authority supported the action. Any subsequent explanation should be traceable to that record.

## 4. Organisational conditions in large enterprises and Indian MSMEs

### 4.1 Adoption conditions in Indian MSMEs

An Indian manufacturing unit, distributor, or service business may concentrate operational decisions in its owner. Responsibility may be identifiable while essential knowledge remains undocumented. Implementation therefore requires the owner and relevant employees to confirm pricing exceptions, credit practices, and supplier arrangements before those rules become executable.

The first implementation task is to establish a usable operating record, with stable product identifiers, reconciled opening balances, and verified customer terms. Informal instructions require review before they become executable rules. Multilingual input can support this work, with explicit validation of interpreted amounts, quantities, and obligations.

Smaller firms may also have limited capacity for data preparation, training, and incident response. A short approval chain does not by itself establish implementation readiness. Adoption should be assessed through the completeness of records, availability of responsible staff, frequency of exceptions, and ability to maintain the system after initial deployment.

For a first workflow, scope should be narrow enough to establish a baseline and resolve exceptions. A distributor might begin with preparation of quotations using verified stock and approved prices, while retaining human confirmation of the final commitment. Further responsibilities can be introduced when the evidence supports them.

### 4.2 Adoption conditions in large enterprises

A large enterprise may already have substantial investment in ERP, specialist applications, and reporting infrastructure. Replacing these systems solely to achieve architectural uniformity can create unnecessary disruption. The initial task is to identify authoritative records and establish where an AI execution layer can interact with them safely.

A regional sales operation may provide a suitable initial scope if customer ownership, inventory access, and financial approvals are clear. Expansion should depend on reconciled results across representative cases, including exceptions and failures, together with evidence that the required controls are operating.

Business units can legitimately operate under different rules. A unified environment must preserve those distinctions while maintaining central controls where required. The implementation question is which definitions should be common, which remain local, and how conflicts are resolved. This requires business ownership as well as engineering work.

## 5. Applied architectural perspective from MatruOS

The MatruOS development context brings CRM, ERP, commerce, institutional knowledge, and agent execution into one business operating environment. The following analysis applies the proposed framework to that

context through design requirements and illustrative workflows. Product performance remains subject to the evaluation requirements in Section 7.

### 5.1 Ownership and portability of operational context

The business should retain usable records of its customers, transactions, policies, and execution history. If a model provider changes, those records should remain accessible and interpretable. Portability requires documented formats, identifiers, policy versions, and export procedures.

Replacing a reasoning model requires evaluation because models can differ in how they interpret instructions, select tools, and handle uncertainty. Representative workflow tests should establish whether the replacement preserves required behaviour. Portable business context supports continuity during that change.

### 5.2 Coordinated execution across business functions

An illustrative MatruOS workflow begins when an enquiry is associated with the correct customer. The system retrieves the applicable commercial terms, checks available stock and credit conditions, and prepares a proposed quotation. Where the request exceeds a permitted threshold, the responsible person reviews it before commitment.

Once the order is accepted, the execution service should reserve the required resources and record the authorised terms. Subsequent invoicing and dispatch depend on their own validated conditions. An external communication is tracked separately so that a failure to send a message does not cause the underlying order to be created again.

The execution record should show which obligations have been fulfilled and which remain open. If stock reservation succeeds but dispatch confirmation is unavailable, the order must retain that unresolved status. This makes shared context useful for the person responsible for completing the transaction.

### 5.3 Cost control within the architecture

The cost of an executed workflow includes model calls, data retrieval, infrastructure, external services, supervision, and correction. An assessment of deployment value also needs implementation and maintenance costs over a stated period, with subscription charges recorded as one part of that total.

For MatruOS, the relevant design principle is to associate resource use with the task and its verified result. An execution that consumes few tokens but requires extensive human correction may be costly. A workflow should therefore record review effort, retries, and exception handling alongside model usage.

Infrastructure choices require assessment against the workload. Serverless execution, connection pooling, and caching each affect resource use under particular operating conditions. A defensible cost model needs the actual configuration, applicable charges, and measured workload.

### 5.4 Establishing value through a bounded deployment

A first deployment should define its starting conditions, permitted actions, acceptance criteria, and responsible owners. Onboarding may assist with initial configuration, but accounting structures, opening stock, permissions, and operational rules require confirmation before consequential execution begins.

A practical pilot could follow one complete quotation and order workflow. Evaluation would record valid quotations, rejected requests, incorrect commitments, manual review, and time to verified completion. Any

projected expansion would then be based on those observations and the additional requirements of the next workflow.

The deployment schedule should reflect data quality, integration work, approval requirements, and staff availability. Progress can be assessed through operational milestones such as reconciled opening records, validated permissions, successful exception handling, and verified completion of the selected workflow.

## 6. Executive governance and continuous operational visibility

The management implications of an AI operating system follow from the availability of connected information and bounded execution. Leadership can define review thresholds and receive relevant exceptions between scheduled meetings. Scheduled reviews still have a role in examining assumptions, persistent failures, and the appropriateness of policy.

### 6.1 Meaningful exceptions and responsible intervention

An alert needs an accountable recipient, evidence, urgency, and a defined response. Repeated notifications without ownership can create additional work. A margin exception should identify the transaction, applicable cost assumptions, threshold, and available action. If the underlying data is incomplete, that uncertainty should be visible.

Operational decisions should remain with the people authorised to make them. Escalation is appropriate when an issue exceeds their authority or when policy cannot resolve the circumstances. Leadership should review recurring exceptions to determine whether the operating rules or allocation of responsibility need revision.

### 6.2 Persistent policies with review and expiry

Recording a decision as a reusable rule can reduce repeated approvals, but it also extends the consequences of that decision. The system should establish when the rule applies, who authorised it, and when it should be reconsidered. Temporary concessions need an expiry condition.

Managers must be able to suspend a policy, revoke delegated authority, or stop a workflow through controls enforced by the application. They also need access to the relevant evidence and sufficient time to intervene before the action becomes irreversible.

### 6.3 Human judgement and organisational responsibility

Relationships, disputed obligations, ambiguous priorities, and changing business conditions require accountable judgement. As routine information handling is automated, a deployment should examine how responsibilities and review demands change for the people involved. Any staffing implications require evidence from the actual operation.

A deployment should also account for employees’ ability to understand, question, and correct system behaviour. Review procedures should capture recurring exceptions and incorporate justified corrections into the operating rules. Otherwise, the same problem may be resolved repeatedly without improving the process.

## 7. Evaluation requirements and limitations

A shared environment creates dependencies on common infrastructure, policies, and access controls. Poor records can propagate incorrect assumptions across functions, while extensive integration can increase the

consequences of a permission error. These conditions limit the proposed benefits and must form part of the evaluation.

Research on agents that use tools provides relevant evaluation methods. τ-bench evaluates interactions involving users, tools, and domain policies by comparing the resulting database state with the intended outcome. It also examines consistency across repeated trials. These methods inform workflow evaluation, while financial attribution requires separate business evidence. [6]

A suitable evaluation of the proposed architecture would compare the same business workflow under documented operating conditions. Measures should include correct final states, unauthorised actions, duplicate commitments, unresolved exceptions, human review time, and total cost per valid completion. Repeated trials should include unavailable services, conflicting records, simultaneous requests, and changed policies.

Financial attribution needs additional care. Time saved should be connected to an observed use of that capacity, such as processing additional demand or reducing administrative backlog. Earnings changes may also reflect prices, demand, staffing, and other process changes. These influences should be recorded rather than attributing every improvement to the AI system.

Security testing must include untrusted content received through tools and documents. AgentDojo demonstrates an evaluation approach for prompt injection in agent environments. An integrated business system needs to test whether such content can redirect an agent beyond the user's intent or permitted authority. [7]

The contribution of this paper is an architectural account of the conditions under which shared context may support reliable business commitments. It contains no new dataset or controlled deployment results. The proposed relationship between integration, coordination effort, and operational value remains open to empirical testing.

## 8. Conclusion

The enterprise AI value gap raises a practical question about how useful individual assistance becomes dependable organisational execution. This paper argues that business context, institutional memory, delegated authority, and transaction controls must be considered together. Their relationship becomes visible when a workflow is followed through to a verified outcome.

A unified AI operating system provides one architectural approach to that problem. Its value depends on authoritative records, applicable policies, bounded permissions, recoverable execution, and accountable oversight. Physical consolidation may support those conditions in some settings; integrated systems of record may support them in others. The appropriate design must reflect the business and its existing obligations.

For Indian MSMEs, implementation requires attention to informal knowledge, operational records, and the capacity to maintain the system. For large enterprises, continuity, local requirements, and integration with established applications remain central. Applying the framework to MatruOS identifies concrete questions for deployment evaluation, including whether policies remain applicable, permissions are enforced, and incomplete commitments are recovered.

The next research task is to measure complete workflows under representative conditions, including exceptions and failures. Evidence of correct execution, controlled authority, and attributable operational value should determine how far responsibility is delegated to the system.

## Author affiliation and disclosure

The author is affiliated with Code Colonies Private Limited, Ahmedabad, and leads the MatruOS team. This professional association is relevant to the architectural discussion of MatruOS. No independent product evaluation is reported.

## References


[1] Tinkoff, D., Van der Veken, L., Chui, M., and Balakrishnan, T. (2026). The state of AI in 2026: On the road to ROI. McKinsey & Company, 25 August.

https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai

[2] Brynjolfsson, E., Rock, D., and Syverson, C. (2021). The Productivity J-Curve: How Intangibles Complement General Purpose Technologies. American Economic Journal: Macroeconomics, 13(1), 333-372.

https://doi.org/10.1257/mac.20180386

[3] Mei, K., et al. (2025). AIOS: LLM Agent Operating System. Conference on Language Modeling. arXiv:2403.16971, version 5.

https://arxiv.org/abs/2403.16971v5

[4] Padhi, A., Elliott, B., Swan, D., Baroudy, K., and Sohoni, V. (2026). The AI operating system: A new way for CEOs to run the enterprise. McKinsey & Company, 13 October.

https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-ai-operating-system-a-new-way-for-ceos-to-run-the-enterprise

[5] PostgreSQL Global Development Group. PostgreSQL 18 Documentation. Section 13.2, Transaction Isolation.

https://www.postgresql.org/docs/18/transaction-iso.html

[6] Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. (2024). τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. arXiv:2406.12045.

https://arxiv.org/abs/2406.12045

[7] Debenedetti, E., et al. (2024). AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. Advances in Neural Information Processing Systems, Datasets and Benchmarks Track.

https://arxiv.org/abs/2406.13352