ETIS TECHNICAL & STRATEGIC WHITE PAPER

Engineering Trustworthy Agentic AI Systems

Governed Autonomy, Intelligent Supervision, and the Protection of Business Trust

William T. O'Connell, Ph.D.

Former IBM Distinguished Engineer and IBM CTO, Quality — Americas | Adjunct Professor of Computer Science, Loyola University Chicago

October 2026 | ETIS Companion White Paper | Technical & Strategic

AGENTS MAY WORK. AGENTS MAY SUPERVISE. NEITHER MAY GOVERN ITSELF.

Executive perspective

Agentic AI is a major business opportunity. Systems that interpret goals, coordinate tools, and perform multi-step work can improve responsiveness, scale operations, and create new services. But the same autonomy creates a new path to business loss: a plausible yet incorrect model inference can become a payment, disclosure, denial, false completion claim, or uncontrolled consumption of enterprise resources—at machine speed.

The answer is not to demand infallible models. It is to engineer governed autonomy. Worker agents perform tasks; supervisory agents observe workflow integrity, behavioral drift, and resource abuse; independent trusted services retain the authority to execute protected actions and commit business state. All layers can fail. The system must therefore employ multiple appropriately independent controls, measured assurance, and recovery.

This is a business-trust architecture, not an AI feature checklist. Customer confidence, regulatory standing, brand equity, and revenue are exposed whenever an agent crosses a business boundary. Executives own the risk and the workforce investment; architects own the design and evidence; emerging engineers must learn to build systems that remain controlled when intelligence is wrong.

The defining question is no longer "Can the agent do the work?" It is "Can the enterprise trust what the agent is allowed to do—and prove it?"

Companion perspective: Becoming a Software Engineer in the AI Era translates these enterprise engineering obligations into five capabilities and a practical evidence-building path for students and emerging professionals.

I | The opportunity—and the obligation

1. What agentic AI changes

The argument builds on accumulated enterprise engineering practice, including the author's experience in IBM Consulting and enterprise technology leadership. It does not claim that the architecture originated with LLMs, or that one vendor's implementations constitute proof of universal effectiveness. Rather, decades of process

management, decision systems, and intelligent automation offer reusable patterns whose assumptions must be tested again as autonomy expands.

Enterprise software has long incorporated statistical inference, machine learning, decision support, and automated workflows. These systems could produce plausible but incorrect recommendations—or technically valid actions that violated business requirements. Modern agentic AI combines more general language models with open-ended goals, tool invocation, iterative planning, and greater operational autonomy. The historical lesson is continuity of engineering obligations; the modern challenge is the enlarged range and speed of consequential actions.

AI-assisted development changes the economics of creating artifacts. AI operating inside the business system changes the locus of proposed decisions and the speed at which consequences can accumulate. Established requirements engineering, transactions, distributed-systems design, security, and operational discipline remain necessary; they now govern a component whose recommendations cannot be assumed correct.

2. Trust is a business obligation, not a model property

A successful tool call can still be a business failure. A refund can be technically processed but exceed eligibility; a claims decision can be procedurally complete but violate a required review; a customer response can be fluent but expose restricted information. Functional execution, business correctness, and customer trust are distinct assurance claims.

Business obligation	Enforceable or evaluable control	Business consequence
Authority and scope	Authenticated identity; action-level permission; approved delegation	Fraud, unauthorized commitments
Business rules	Eligibility predicates; thresholds; state-machine guards	Revenue leakage, disputes
Privacy and compliance	Purpose limitation; access restrictions; required review	Penalties, customer harm
Fairness and treatment	Outcome evaluation; appeals; human accountability	Disparate harm, brand erosion
Resilience and repair	Idempotency; containment; reconciliation; incident response	Service disruption, irrecoverable effects

Some obligations can be encoded as hard invariants; others—including fairness, appropriate customer treatment, and explanation quality—require evaluation, monitoring, escalation, and accountable judgment. A policy engine can deny a prohibited payment. It cannot by itself establish that every outcome is ethical. The architecture must support both kinds of assurance. This aligns with lifecycle-oriented risk governance described by NIST [1, 2].

3. Autonomy must be an explicit executive choice

Executives should recognize the continuity with existing automation estates. Prior investments in BPM, business rules, RPA, service authorization, audit, and reliability engineering may be reusable assets. They are not automatic certification for agentic workloads: leaders should fund a gap assessment that tests whether these controls remain non-bypassable when agents dynamically choose actions and coordinate with one another.

Every agentic workload needs a defined authority envelope: what goals it may pursue, which resources it may use, which actions it may request, what loss it may cause, when it must stop, and who accepts residual risk. “Deploy the agent” is not an adequate business decision. Leaders must approve the value proposition together with the risk, controls, operating cost, and workforce capability required to sustain it.

The more autonomy an enterprise grants, the more deliberately it must engineer the boundaries of that autonomy.

II | The architecture of governed autonomy

Established foundations: probabilistic intelligence, control, and enterprise workflows

The architectural challenge predates large language models. Machine-learning classifiers, predictive scoring, and statistical decision support have long produced useful but fallible inferences: a prediction can be statistically plausible yet factually wrong, affected by data or concept drift, or incompatible with eligibility, privacy, or contractual rules. Even a correct prediction does not establish permission to act. Enterprise engineering therefore distinguishes model quality from business correctness, authorized execution, and verified outcomes. This paper applies that established distinction to agentic AI, rather than claiming to originate it [1, 14].

Complementary predecessors contributed different controls. Robotics and autonomous-control research emphasized feedback, bounded action, and response to uncertain observations; these are relevant analogies, not claims that all robots used probabilistic models [14]. Business Process Management (BPM), rules engines, and transactional services separated workflow coordination from authorized execution. IBM's ILOG and Lombardi acquisitions illustrate business-rule and process-management investment; its 2020 WDG Automation acquisition added robotic process automation (RPA) bots [9–11]. Conventional RPA is generally scripted, not inherently probabilistic. When combined with classification, natural-language understanding, or adaptive decisions, however, uncertain outputs can influence business workflows. IBM Research also documented conversational, multi-agent orchestration with RPA [12].

The reusable patterns are specific: keep business authority outside probabilistic assertions; validate eligibility and authorization at the action boundary; preserve durable, authoritative workflow state; distinguish successful technical execution from a valid business outcome; monitor changing model behavior; and reconcile, contain, or compensate for partial failures. These patterns arose across several disciplines, not from one vendor or one technology. Modern LLM-based agents extend the problem through open-ended planning, dynamic tool selection, multi-agent coordination, changing context, and machine-speed propagation. Reuse the patterns, but test their assumptions against the enlarged action space and new attack paths [14].

4. Four responsibilities—not four levels of trust

A model (LLM or SLM) produces inferences. A worker agent combines inference with goals, workflow logic, tool requests, and progress reports. A supervisory agent evaluates events and recommends verification or containment. A trusted control plane, supported by protected business services, holds authoritative state and independently enforces business policy, identity, authorization, resource limits, and allowed transitions. Coordination may schedule agents, but scheduling is not business authority.

Supervisory agents are not a privileged class of intelligence. They are fallible, may be manipulated, and must operate through the same independently enforced boundaries as workers. These are logical roles; implementations may use multiple services, models, or deterministic monitors without changing the underlying trust requirement.

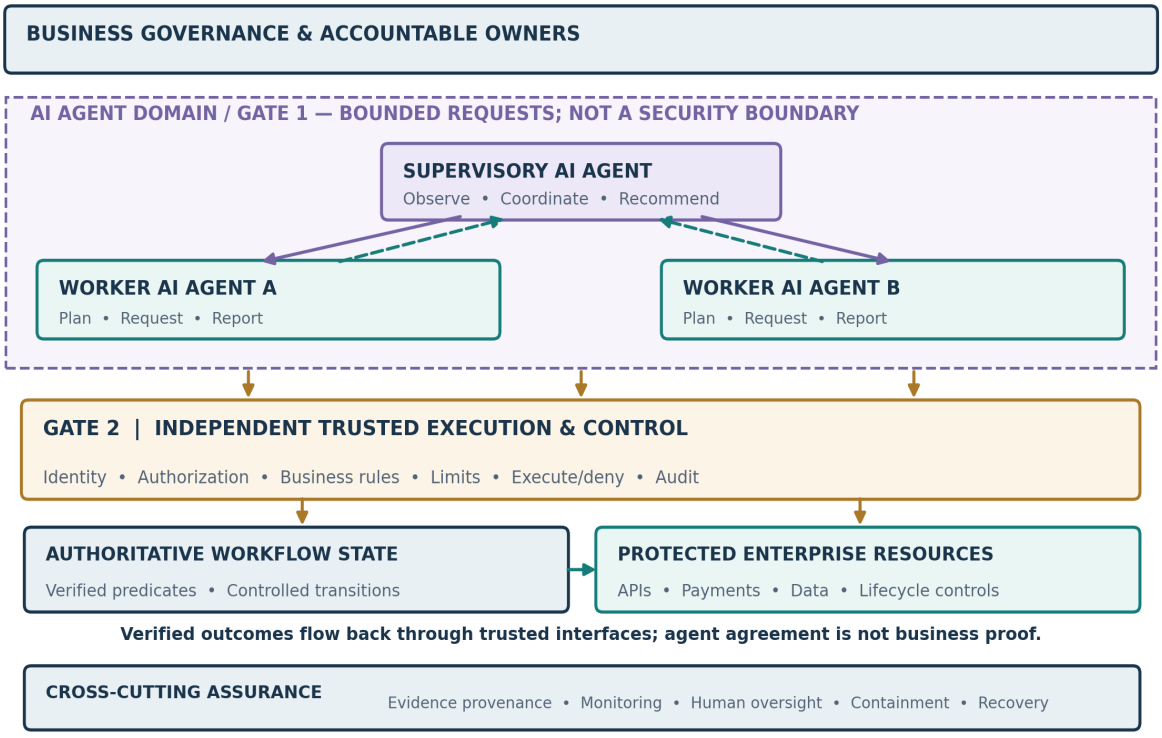

Figure 1. ETIS reference architecture. Supervisory and worker AI agents coordinate inside a logical, non-privileged domain. All consequential operations require independent trusted enforcement and authoritative business evidence. Dashed arrows indicate status feedback, not delegated security authority.

A worker AI agent is a goal-directed software component that uses model inference and orchestration to propose and request business work. A supervisory AI agent observes workers and the operating environment to recommend interventions. Both roles may use an LLM or SLM, but neither role possesses inherent authority over protected enterprise resources. The logical boundary around agents is not a security boundary.

5. The mandatory execution boundary

Gate 1 belongs to the agent application: input checks, bounded context, task logic, retries, and normal error handling. Gate 2 belongs to independent trusted execution: authenticated workload identity, business preconditions, authorization, transaction ceilings, approval requirements, authoritative state changes, and durable evidence. A prompt is not an authorization boundary. The agent must not be able to bypass Gate 2 through its permitted tools.

The supervisory AI agent may coordinate worker agents directly for ordinary task assignments and status exchange. Such communication cannot authorize payments, certify completion, change privileges, or suspend another agent. Privileged interventions—whether requested by a worker or recommended by a supervisor—must cross the same trusted enforcement boundary. The supervisor has coordination authority, not security authority.

For protected databases, files, networks, and consequential APIs, the preferred pattern is that the agent does not possess unrestricted resource credentials. It submits a narrowly specified operation to a trusted service, which holds or obtains the necessary resource authority and evaluates policy before execution. Short-lived delegated credentials can be justified for limited cases, but narrower duration does not guarantee business correctness; even a brief authorized action may be irreversible. Delegation must therefore be risk-assessed, constrained at the resource, observable, and revocable.

Gate 2 is not itself a guarantee of correctness. The protected resource must retain its own least-privilege and transaction safeguards; control services require separation of duties, change governance, monitoring, and tested recovery. Where two controls depend on the same faulty policy code, apparent redundancy may not prevent a common-mode failure. The goal is layered containment, not an impossible promise of bug-free software.

Consider a customer-service agent that proposes a $100,000 refund on a $100 order. The trusted transaction service checks customer binding, order eligibility, permissible amount, duplicate status, and approval conditions. The excess is denied before payment. The agent may be repaired later; the independent business control protects the transaction now.

6. Completion is an authoritative state transition

The state store is an enterprise authority, not an agent memory. Trusted workflow services own its transitions; agent-generated narratives, tool output, and supervisor opinions are inputs to evaluation, not proofs of business completion. When a required approval or settlement record is absent, the workflow must remain blocked regardless of agent consensus.

A worker may announce “refund complete,” and a supervisor may agree. Neither assertion makes the workflow complete. A trusted workflow service must evaluate the actual business predicates—authorized amount, payment settlement, required notifications, reconciled records, and retained evidence—before committing COMPLETED. Otherwise it must retain PENDING, HOLD, FAILED, or NEEDS_REVIEW as appropriate. Supervisors can challenge missing evidence; they cannot manufacture the authoritative outcome.

Evidence provenance: agreement is not verification

A worker can report a successful settlement; a supervisor can repeat the claim; and an audit log can faithfully preserve both statements. None proves the payment settled. For consequential state changes, trusted services must obtain evidence from an authoritative source—for example, an authenticated payment-provider record—and preserve the link between that source, the evaluated policy, and the resulting transition. Repetition of an unsupported assertion by several agents is not independent corroboration. The ETIS distinction is between a record of what agents said and evidence of what the business actually did. Prompt-injection research reinforces the need to treat externally supplied text as untrusted [6, 7].

A useful parallel comes from distributed ledgers. Blockchain consensus can establish agreement about a recorded ledger state, but it cannot independently establish the truth of external inputs—the “oracle problem” documented by NIST [13]. Similarly, agreement between worker and supervisory agents, or a tamper-evident log of their claims, does not establish that a refund settled. This is an analogy about evidence authority, not a proposal to use blockchain as an agentic workflow database.

This distinction matters in partial failure. An API can acknowledge a request while the payment remains unsettled; retries can duplicate effects; a model can lose context and claim success. Durable state, idempotency keys, transactional outbox patterns where appropriate, reconciliation, and compensating operations protect against these conditions. A kill switch stops new work; it does not undo a committed payment.

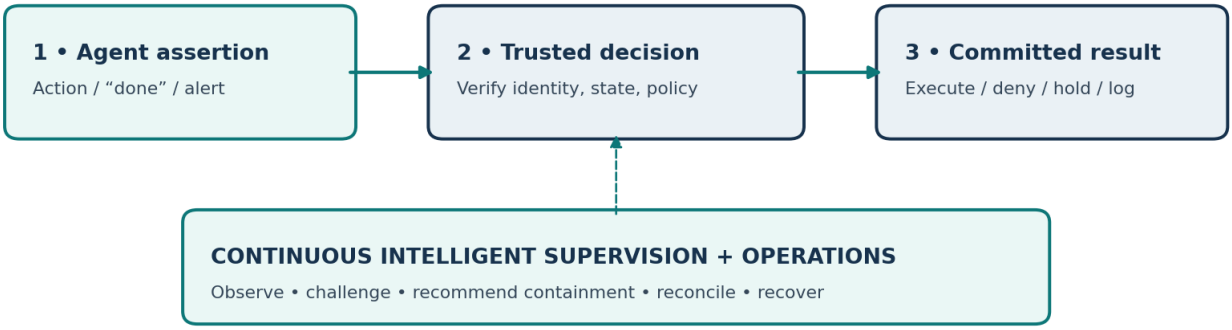

Figure 2. A recommendation or completion claim becomes an enterprise outcome only after trusted validation.

7. Intelligent supervision without supervisory sovereignty

Human operators cannot inspect every tool call, status transition, or hostile request at machine speed. Supervisory intelligence can help interpret complex signals and recommend response. Three complementary functions deserve explicit design:

Supervisory function	Observed signal	Possible recommendation
Workflow assurance	Missing approvals, unsupported completion, stale evidence	Hold transition; require reconciliation
Behavioral assurance	Goal deviation, unusual tool sequence, prompt-injection indicators, repeated failures	Reduce permissions; investigate; isolate
Resource defense	External agent swarms, API abuse, budget exhaustion, suspicious access	Throttle; quarantine; block; escalate

Machine-speed defense must also be bounded by machine-enforced policy. Supervisory agents may recommend throttling hostile requests, suspending a runaway worker, or isolating a compromised session, but trusted services must decide and execute those interventions within preauthorized limits. An attacker can poison telemetry or imitate a legitimate worker; therefore observation provenance, identity checks, rate limits, and safe escalation are required for defensive decisions as well as business transactions.

Supervision is not infallible. Drift indicators are not proof of wrongdoing, and a malicious input may attempt to deceive the supervisor. Low-impact reversible containment may be preauthorized under trusted policy; high-impact actions need stronger checks or accountable human approval. Preventive limits must remain effective even if the supervisor is unavailable or mistaken.

Supervision itself expands the attack surface

Supervisory agents consume traces, tool responses, messages, incident reports, and other operational signals. Those inputs may include attacker-controlled text. An adversary can attempt to forge a worker status, poison a monitoring summary, or insert instructions that induce the supervisor to ignore abuse or disable legitimate operations. Research on tool-using agents demonstrates that untrusted tool outputs can redirect agent behavior [6]; OWASP separately identifies inter-agent communication, memory and context poisoning, and cascading failures as agentic risks [4]. The ETIS inference is direct: a supervisory model cannot be treated as an authoritative security monitor simply because its purpose is defensive.

Design the supervisor as a bounded observer. Authenticate event producers where possible; preserve provenance and integrity of telemetry; separate observed data from instructions; restrict tools and delegated powers; and require trusted policy to approve consequential interventions. Test not only whether a supervisor catches malicious behavior, but whether a malicious worker or external actor can manipulate the supervisor into blocking a healthy workflow. A compromised supervisor must not become the shortest path to compromising the enterprise.

8. The system must protect its own resources

A production agentic system cannot wait for an external security operations team to notice every abuse. It must enforce workload identity, admission control, concurrency and spending budgets, data-access policy, segmentation, quotas, rate limits, and containment on its own execution paths. Supervisory resource-defense agents add adaptive pattern recognition, including against coordinated hostile external agents. The enforcement decision remains with trusted infrastructure and services.

This is not a replacement for enterprise cybersecurity. Corporate security provides cross-system standards, intelligence, incident coordination, and independent oversight. The application architecture must implement its own operational self-protection; security cannot be outsourced to a policy document or an after-the-fact alert. OWASP identifies excessive agency, prompt injection, and unbounded consumption as relevant application risks [3, 4].

9. Who governs the trusted control plane?

The trusted layer is engineered software, not a magical source of correctness. It can contain defects, be misconfigured, lose availability, or be compromised. The answer is not an endless hierarchy of AI supervisors. It is layered, independently testable engineering: service-level enforcement downstream of gateways; separation of duties; scoped service identities; protected change processes; policy review; negative testing; telemetry; audit; and exercised recovery. Where two gates reuse the same defective policy code or shared configuration, apparent redundancy may offer little protection.

Controls should be independent in failure mode, not merely numerous. For example, a gateway may approve a transaction incorrectly while a downstream payment service independently rejects amounts above a fixed limit. A monitoring system may flag suspicious aggregate behavior. These controls reduce risk, but cannot promise zero defects or zero loss. The obligation is to make credible exposure bounded where feasible, detectable, recoverable, and accountable.

Engineering does not guarantee that every component is right. It prevents one component being wrong from automatically becoming an uncontrolled business failure.

10. Established engineering patterns still matter

Earlier predictive systems could recommend an action that was statistically reasonable yet commercially impermissible; workflow engines and robotic process automation (RPA) could then carry out a specified operation correctly while violating a business obligation if the surrounding controls were deficient. Today's agents can generate and revise plans, select unfamiliar tool combinations, and coordinate with other agents. The enduring question is whether a plausible inference is allowed to become a consequential action—and whether apparent completion is supported by authoritative business evidence. What changes is the range of possible trajectories, their speed, and the assurance required to govern them.

Agentic architecture inherits the economics of distributed computing: data locality, queueing, latency, consistency, failure isolation, observability, and recovery. Parallel database systems established the value of pushing suitable computation toward large distributed datasets. In an enterprise with millions of customer

records, SQL, streaming analytics, and bounded local inference can reduce data before an agent or central model synthesizes results. Sending raw data to a large model is not a substitute for database architecture.

Centralized, distributed, and hybrid designs each have merits. Engineers must compare complete systems against throughput, inference cost, network movement, data residency, provenance, control complexity, and service objectives. Teradata-era compute-to-data is one instructive historical pattern, not a universal mandate. The deeper lesson is to reuse proven principles rather than mistake a new model for a new exemption from engineering.

III | Trust must be demonstrated and sustained

11. War-game the business system—not just the model

Agent benchmarks can illuminate capability, but they are not enterprise assurance cases. Kapoor and colleagues show why evaluation must account for cost, reproducibility, and holdout quality rather than report task accuracy alone [8]. AgentDojo supplies a concrete example of testing tool-using agents against adversarial inputs [6]. Neither result validates an individual enterprise deployment; both motivate more disciplined, application-specific evaluation.

A demonstration proves that a selected path worked under observed conditions. A trustworthy deployment needs an assurance case linking each material business obligation to a control, accountable owner, negative test, acceptance threshold, operational signal, and recovery mechanism. This extends secure-development discipline [5] and lifecycle AI risk governance [1, 2] to agentic execution.

Adversarial test	Required system behavior	Evidence
Worker claims completion before settlement	Reject COMPLETED; preserve authoritative pending state	Transition denial and reconciliation trace
Worker requests excess refund	Reject before transfer; record policy decision	Negative authorization test
Supervisor falsely recommends suspension	Bound or deny action; provide review and restoration	Decision record; false-positive recovery
Forged monitoring event redirects supervisor	Reject unauthorized intervention; preserve event provenance	Injected-event test; source and policy trace
Worker and supervisor repeat false settlement	Reject completion absent independent settlement evidence	Authenticated source record; state guard trace
External agent swarm exhausts resources	Throttle/isolate while protecting critical services	Load, budget, and containment metrics
Gateway defect permits unsafe action	Downstream independent limit denies or contains	Fault injection; service-level trace
Partial commit followed by retry	Avoid duplicate payment; reconcile outcome	Idempotency and compensation evidence
Model/policy/tool update	Reassess affected controls and assurance claims	Versioned regression and approval record

12. Machine-speed detection is not machine-speed certainty

Anomalies can be detected too late; false positives can harm customers; malicious events can corrupt telemetry. Synchronous preventive checks protect hard invariants, while adaptive supervision observes patterns

and recommends responses. Define and measure detection delay, containment time, credible loss, service degradation, false-positive cost, and recoverability. A high-risk money-moving agent demands stronger assurance than a read-only internal summarizer.

Human oversight must be meaningful rather than ceremonial: an approver needs sufficient context, time, authority, and a real opportunity to stop the action. When action is too fast for human review, preauthorized trusted limits and reversible containment are essential. Human accountability does not imply human reaction speed for every event.

13. Assurance must be economically sustainable

Governance has a cost: policy decisions, additional trusted services, supervisor inference, storage of evidence, incident review, human escalation, and false-positive disruption. A design that minimizes technical risk but makes the service uneconomic may fail the business objective. Agent evaluation research specifically warns against treating accuracy independently of cost [8]. For ETIS, the architectural consequence is to evaluate business value, residual loss exposure, and the total cost of assurance together—not to optimize any one in isolation.

A practical decision compares the expected business value of autonomy against inference and tool costs, enforcement and observability overhead, latency, customer harm from mistaken interventions, and the cost of human review. Critical financial actions may justify independent downstream authorization and continuous transaction monitoring; a read-only internal assistant may not. Risk-proportionate governance is not weaker governance: it is governance engineered to the consequence, reversibility, and scale of each workload.

The objective is not maximum autonomy or maximum control. It is economically sustainable autonomy within an explicitly accepted risk boundary.

14. Governance is an operational lifecycle

Prompts, model versions, tools, data, permissions, policies, and business rules change. Each change can invalidate earlier assurance. Operational governance therefore requires versioned configurations, controlled releases, monitoring, audit trails, drift evaluation, incident learning, and periodic reassessment of residual risk. The system must answer: Which rule applied? Who had authority? What evidence supported the transition? Which control acted? What changed? How was a failure contained and repaired?

A trustworthy agentic system does not merely make decisions. It can demonstrate why its consequential actions were permitted, which authoritative evidence justified them, and what happened afterward.

IV | The engineering judgment and workforce imperative

15. Architecture determines the talent strategy

This continuity changes the talent question. Experienced engineers bring transferable expertise in workflow engines, decision management, distributed transactions, API security, and production operations; emerging engineers need those foundations alongside model evaluation and adversarial testing. An enterprise that treats agentic AI as wholly new risks discarding hard-won capability. An enterprise that treats it as merely RPA with better language understanding risks underestimating autonomy and attack surfaces.

The difficulty is not simply finding people who can build an agent demo. It is developing engineers who can translate business obligations into governed execution, recognize distributed-systems trade-offs, test hostile paths, operate systems under failure, and preserve evidence. Practitioner experience suggests these combined capabilities can be scarce; this paper does not treat that observation as a quantified labor-market finding.

AI-assisted code generation may reduce implementation effort while raising the importance of engineering judgment. Executives must therefore treat talent composition as part of the agentic architecture investment—not as an independent human-resources exercise. Experienced engineers who adapt bring valuable historical pattern recognition; newer engineers may leapfrog in emerging specializations without acquiring decades of operational experience overnight.

16. Responsible workforce transformation includes difficult decisions

Major technology transitions require organizations to reassess capability, investment, and staffing. Leadership should define the future skills profile; provide meaningful reskilling, mentoring, assignments, and incentives; recognize rapid adopters; and redeploy people into work where they can contribute. These actions are good for employees who adapt and for the business that needs their capabilities.

Not every role or skill mix will remain viable. When reasonable development and reassignment options do not close a sustained capability gap—or when strategy eliminates a role—workforce reductions or employment separation may be necessary. Those decisions should be fair, evidence-based, consistent with applicable obligations, and not framed as a moral judgment about displaced employees. Avoiding every difficult decision can weaken competitiveness; using layoffs as a substitute for coherent workforce planning can do the same.

A credible agentic AI strategy funds both the control architecture and the workforce transition required to operate it.

17. Developing the next generation is a three-party obligation

Employers must preserve apprenticeship through bounded ownership, design reviews, incident participation, recovery exercises, and progressive responsibility. Universities must retain computing fundamentals while assessing design reasoning, testing, collaboration, and individual evidence—not merely finished AI-assisted artifacts. Students must engage, ask questions, surface uncertainty early, and learn to defend decisions. If AI removes some traditional entry-level practice, organizations must redesign that practice rather than quietly eliminate the experience ladder.

Professional collaboration is itself an operational control. Clear ownership, dependencies, checkpoints, phase gates, and early escalation prevent hidden problems from propagating. Not knowing is normal; not learning is a problem; not communicating uncertainty that affects others is a team problem.

18. Five dimensions of engineering potential

The capabilities below are derived from the demands of trustworthy agentic systems. They form an ETIS professional-development framework, not a validated psychometric hiring instrument. The companion ETIS educational white paper, “Becoming an Engineer in the AI Era: Five Dimensions of Professional Potential and the Opportunity to Leapfrog,” develops the student implications in greater depth.

Engineering requirements determine capability needs—not the reverse.

Figure 3. The requirements of trustworthy systems lead to organizational investment and five human capabilities.

Dimension	Why it matters	Concrete demonstration
1. Technical foundations	Understand models, distributed systems, identity, state and transactions	Diagram an agent-to-service trust boundary
2. Engineering judgment	Choose trade-offs; preserve business invariants; plan for failure	Defend authorization and recovery design
3. Learning agility	Adapt established principles to new models and attack modes	Revise an architecture after adversarial findings
4. Collaboration	Surface risks, coordinate dependencies, uphold commitments	Show reviews, escalation, and team evidence
5. Initiative and evidence	Take ownership and make claims independently reviewable	Deliver negative tests, audit traces, and recovery drills

For entry-level hiring, technical fundamentals establish a threshold; reasoning, learning agility, and collaboration often differentiate how candidates confront unfamiliar work; initiative and evidence make those capabilities observable. Employers must then deepen all five through real experience. A degree is an entry credential, not proof of readiness to govern a production agentic system.

19. Decisions each audience must make now

Executive	Architect / engineer	Student / emerging engineer
Approve autonomy with explicit authority, loss tolerance, and accountability.	Design trusted execution, independent downstream controls, and authoritative state.	Learn to explain model, agent, service, and authority boundaries.
Fund controls, incident recovery, and workforce reskilling/reallocation.	Test worker, supervisor, control-plane, and external-attack failures.	Build portfolios of trade-offs, negative tests, and recoverability.
Measure customer trust and residual exposure—not agent count or code volume.	Preserve evidence; monitor drift; rehearse containment and recovery.	Demonstrate learning agility, teamwork, and early escalation.

Conclusion | Intelligence is not authority. Autonomy is not accountability.

Agentic AI creates a consequential new business capability: software that interprets goals, coordinates work, and acts. It also creates a consequential new failure path: probabilistic decisions can become business actions at machine speed. The enterprise cannot transfer its obligations to a model, a worker agent, or a supervisory agent. The obligation to protect customers, revenue, compliance, and brand trust remains with the organization.

For executives: change the investment decision. Stop asking only how much work agents can automate. Demand a credible account of what they are allowed to do, how failure is bounded, who owns residual risk, and whether the workforce can operate the architecture. Invest in reskilling and redeployment—and make fair, difficult resource-allocation decisions when necessary. Business trust is not an afterthought; it is an asset the architecture must protect.

For architects and professional engineers: change the acceptance standard. An impressive agent demonstration is not a trustworthy system. Assume workers, supervisors, and trusted services can fail. Keep consequential authority and completion state outside probabilistic agents. Design multiple independent controls, challenge

their shared failure modes, test adverse conditions, and demonstrate containment and recovery. Governance is architecture, and evidence is the price of trust.

For students and emerging engineers: change what you prove. Generating an agent is a starting point, not a professional achievement by itself. Learn the foundations, explain architectural trade-offs, work reliably with others, adapt quickly, and show reviewable evidence that your system remains governed when its intelligence is wrong. That is the higher bar—and the opportunity—of the agentic era.

The competitive advantage will not belong simply to enterprises with the most autonomous agents. It will belong to enterprises that can deploy autonomy without surrendering business trust.

AGENTS PERFORM THE WORK. AGENTS SUPERVISE THE WORK. TRUSTED ENGINEERING GOVERNS THE SYSTEM.

Scope and architectural assumptions

Historical IBM product and acquisition descriptions document predecessors and their intended functions, not outcomes from particular consulting engagements. IBM's overview of agent evolution provides historical context, not independent proof that all earlier systems shared modern agentic failure modes [14]. The continuity argued here is a practitioner-informed synthesis drawing on experience reviewing enterprise automation and machine-learning systems. Scripted RPA is not equated with probabilistic intelligence; robotics is used as a bounded-control analogy. Historical lessons guide architecture but cannot substitute for contemporary workload-specific evidence.

This paper presents a conceptual enterprise reference architecture and professional thesis. Its diagrams show logical responsibilities, not a single prescribed deployment topology. Independence, non-bypassability, and loss containment are design objectives that must be verified against concrete infrastructure, threat models, business policies, and failure modes. No claim of defect-free software, universal ethical correctness, or zero residual risk is made. Legal and regulatory requirements vary by jurisdiction and use case. The companion ETIS education paper describes how students and early-career engineers can develop the underlying capabilities.

References

[1] Tabassi, E. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1. https://doi.org/10.6028/NIST.AI.100-1

[2] Autio, C., et al. (2024). Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. NIST AI 600-1. https://doi.org/10.6028/NIST.AI.600-1

[3] OWASP GenAI Security Project. (2025). Top 10 for Large Language Model Applications (including prompt injection, excessive agency, and unbounded consumption). https://genai.owasp.org/llm-top-10/

[4] OWASP GenAI Security Project. (2025). OWASP Top 10 for Agentic Applications. https://genai.owasp.org/2025/12/09/owasp-genai-security-project-releases-top-10-risks-and-mitigations-for-agentic-ai-security/

[5] Souppaya, M., Scarfone, K., & Dodson, D. (2022). Secure Software Development Framework (SSDF) Version 1.1. NIST SP 800-218. https://doi.org/10.6028/NIST.SP.800-218

[6] 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 37, Datasets and Benchmarks

Track. https://proceedings.neurips.cc/paper_files/paper/2024/file/97091a5177d8dc64b1da8bf3e1f6fb54-Paper-Datasets_and_Benchmarks_Track.pdf

[7] Jia, F., Wu, T., Qin, X., & Squicciarini, A. (2024). The Task Shield: Enforcing Task Alignment to Defend Against Indirect Prompt Injection in LLM Agents. arXiv:2412.16682. https://arxiv.org/abs/2412.16682

[8] Kapoor, S., Stroebl, B., Siegel, Z. S., Nadgir, N., & Narayanan, A. (2024). AI Agents That Matter. arXiv:2407.01502. https://arxiv.org/abs/2407.01502

[9] IBM. (2008). IBM Announces Plans to Acquire ILOG. SEC-filed announcement. https://www.sec.gov/Archives/edgar/data/1031140/000095015708000538/ex99-a5a.htm

[10] IBM. (2010). IBM Closes Acquisition of Lombardi Software. https://www.prnewswire.com/news-releases/ibm-closes-acquisition-of-lombardi-software-82673062.html

[11] IBM. (2020). IBM to Acquire WDG Automation to Advance AI-Infused Automation Capabilities for Enterprises. https://www.prnewswire.com/news-releases/ibm-to-acquire-wdg-automation-to-advance-ai-infused-automation-capabilities-for-enterprises-301090011.html

[12] Rizk, Y., Isahagian, V., Boag, S., Khazaeni, Y., Unuvar, M., Muthusamy, V., & Khalaf, R. (2020). A Conversational Digital Assistant for Intelligent Process Automation. Business Process Management (BPM 2020). https://research.ibm.com/publications/a-conversational-digital-assistant-for-intelligent-process-automation

[13] Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2018). Blockchain Technology Overview. NISTIR 8202, Section 7.3. https://doi.org/10.6028/NIST.IR.8202

[14] Stryker, C. (n.d.). The Evolution of AI Agents. IBM Think. https://www.ibm.com/think/topics/evolution-of-ai-agents