ARCHITECTURE 2.0

happiest minds
AI FIRST. AGILE ALWAYS.

Software with Agentic AI

How to Re-architect Applications to Evolve into AI Applications with AI/Agentic AI

Table of CONTENTS

Executive Summary

Application architecture is undergoing its most significant transformation since the shift from monoliths to microservices. Agentic AI — the paradigm of building systems where AI agents reason, plan, act, and adapt to achieve complex goals — is introducing a cognitive layer on top of decades of architectural evolution. However, this is not about replacing everything with AI. The architecture should follow the problem, not the trend. This whitepaper provides a practical framework for understanding the paradigm shift, evaluating application portfolios for AI readiness, selecting hybrid architecture patterns, and addressing security and testing challenges unique to agentic systems.

The Paradigm Shift

The history of application architecture follows a consistent pattern of decomposition into increasingly intelligent components: monoliths (1990s), SOA (2000s), microservices (2010s), and now agentic AI (2020s+). The crucial distinction: microservices decompose code for a better development lifecycle, while agentic AI decomposes tasks and goals for autonomous intelligent execution. They solve fundamentally different problems — agentic AI builds on top of microservices rather than replacing them.

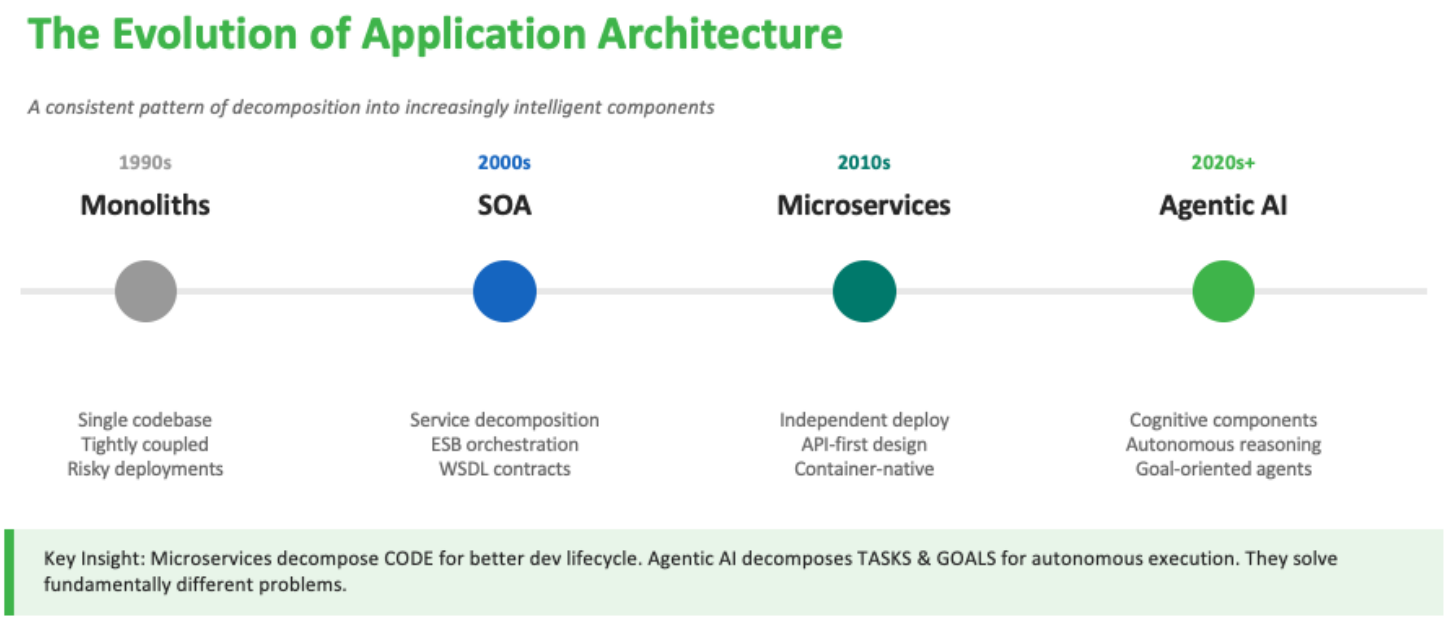

Figure 1: The Evolution of Application Architecture — from monoliths to agentic AI

Defining AI Agents and Agentic AI

- **Traditional Software**: Executes predefined logic. Every step hardcoded. No deviation, no judgment.
- **Generative AI (LLMs)**: Creates human-like content on request but is fundamentally reactive — cannot act, call APIs, or trigger workflows.
- **AI Agent:** Uses an LLM as its reasoning core but goes beyond text — perceives its environment, reasons about what to do, acts on the real world through tools, and adapts based on outcomes.

A traditional application follows a script.
An AI agent writes its own script at runtime to achieve a goal.

The business logic does not disappear in agentic systems — it changes form. Rules expressed as if/else code become policies, playbooks, and contextual knowledge that the agent reasons over. The rules still need to be written and maintained, but can be expressed in natural language, updated without code deployments, and handle combinatorial complexity impractical to encode as explicit decision trees.

Six Fundamental Architectural Shifts

01 **Deterministic → Probabilistic**
Components that reason, not just execute predefined logic

02 **API Contracts → Capability Discovery**
Agents advertise capabilities via Agent Cards; dynamic binding

03 **REST/gRPC → MCP + A2A + ACP**
New protocol stack: agent-to-tool (MCP), agent-to-agent (A2A)

04 **Orchestration → Goal-Oriented Autonomy**
Agents advertise capabilities via Agent Cards; dynamic binding

05 **External State → Memory as Architecture**
Short-term, long-term, and shared memory are first-class concerns

06 **Event-Driven → Structurally Essential**
Loose coupling for agents; but this time mandatory not optional

Deterministic to Probabilistic: Traditional components execute predefined logic; AI components reason and compose novel paths. Runtime dynamism exists in traditional systems (Java reflection, Spring DI) but operates within bounded rules. Agentic systems exhibit emergent reasoning.

API Contracts to Capability Discovery: Agents advertise capabilities through Agent Cards; other agents discover them dynamically, making selection a semantic decision rather than URL-based routing.

REST/gRPC to MCP + A2A + ACP: MCP (Anthropic) standardizes agent-to-tool communication; A2A (Google/Linux Foundation) enables peer-to-peer agent communication; ACP (IBM) provides lightweight HTTP-based messaging.

Orchestration to Goal-Oriented Autonomy: Traditional engines execute developer-designed workflows. Agents compose workflows at runtime using the ReAct pattern: reason, act, observe, adjust.

External State to Memory as Architecture: Memory becomes a first-class component: short-term (task context), long-term (learned patterns), and shared (multi-agent coordination).

Event-Driven becomes Structurally Essential: In microservices, EDA was a design preference. In agentic systems, it is a prerequisite — agents have unbounded execution times and compose workflows dynamically, making synchronous chains unworkable.

The Decision Framework

Not every application needs AI. AI belongs where human judgment currently sits, not where deterministic logic already works well.

When AI is Not Justified

If a problem is fully specifiable and the decision tree is bounded, traditional architecture will outperform AI — faster, cheaper, more auditable. Examples: payroll calculations, SWIFT message routing, inventory deductions, flight booking engines.

When AI Genuinely Adds Value

- **Unstructured Inputs**:
  Free-form text, images, voice, mixed-media documents
- **Judgment Under Uncertainty**:
  Decision space too large for a rule engine to enumerate
- **Cross-Domain Synthesis**:
  Reasoning across information from multiple domains
- **Continuous Adaptation**:
  Rules shift frequently; re-engineering logic is expensive
- **Long-Tail Edge Cases**:
  80% follows clean paths; 20% consumes disproportionate human effort

The ROI Test: Calculate what you spend on human labor bridging the gap between deterministic capability and real-world demand. That gap is where AI investment is justified.

When Human Intervention is Non-negotiable

Even in fully agentic architectures, six scenarios require humans by design: high-stakes irreversible decisions, regulatory and legal mandates, novel situations outside known patterns, ethical and reputational judgment, agent disagreement or conflicting signals, and low confidence on high-value cases.

**Human-in-the-loop is not a weakness in architecture — it is a design feature.
The best agentic systems know their own boundaries.**

Evaluating Application Portfolios

Systematically identifying which applications to augment with AI requires a Cognitive Gap Analysis across five dimensions: Input Variability (D1), Decision Complexity (D2), Data Richness (D3), Human Effort Concentration (D4), and Error Tolerance (D5).

The Cognitive Gap Analysis: Five Dimensions

D1 — Input Variability
How structured/predictable are inputs? Fully structured → low AI need. Unstructured → high AI potential.

D2 — Decision Complexity
Lookup/calc → deterministic. Judgment-dependent or currently outsourced to humans → strongest AI signal.

D3 — Data Richness
Historical decisions with outcomes? Cross-domain correlations? Temporal patterns? Rich data = higher readiness.

D4 — Human Effort Concentration
Where do humans spend time? Data entry, triage, investigation, review, communication — each maps to AI capability.

D5 — Error Tolerance & Auditability
Zero tolerance → AI as copilot only. Higher tolerance → more AI autonomy. Determines HOW AI is introduced.

Scoring applications across these dimensions clusters them into four categories:

Category	Characteristics	Action
Don't change	Low variability, lookup/calc decisions, structured data	**Keep Deterministic**. Examples: payroll, SWIFT
Augment at Edges	Mostly deterministic with specific human intervention points	**Sidecar pattern.** Examples: onboarding, help desk
Cognitive Overlay	Significant human effort in interpretation and judgment	**Cognitive Middleware.** Examples: claims, underwriting
AI-Native Rebuild	Entire value proposition is intelligence-driven	**Multi-Agent Mesh.** Examples: fraud detection, pricing

Hybrid Architecture Patterns

The evolved system is not everything-is-agents — it is a selective cognitive overlay on a solid deterministic foundation or a AI side car addition without disturbing existing application.

Example: Traditional Insurance Claims

Insurance claims processing illustrates the hybrid reality. The traditional multi-tier architecture handles straightforward claims but routes complex cases to human adjusters — creating a bottleneck of hundreds of FTEs doing cognitive manual work.

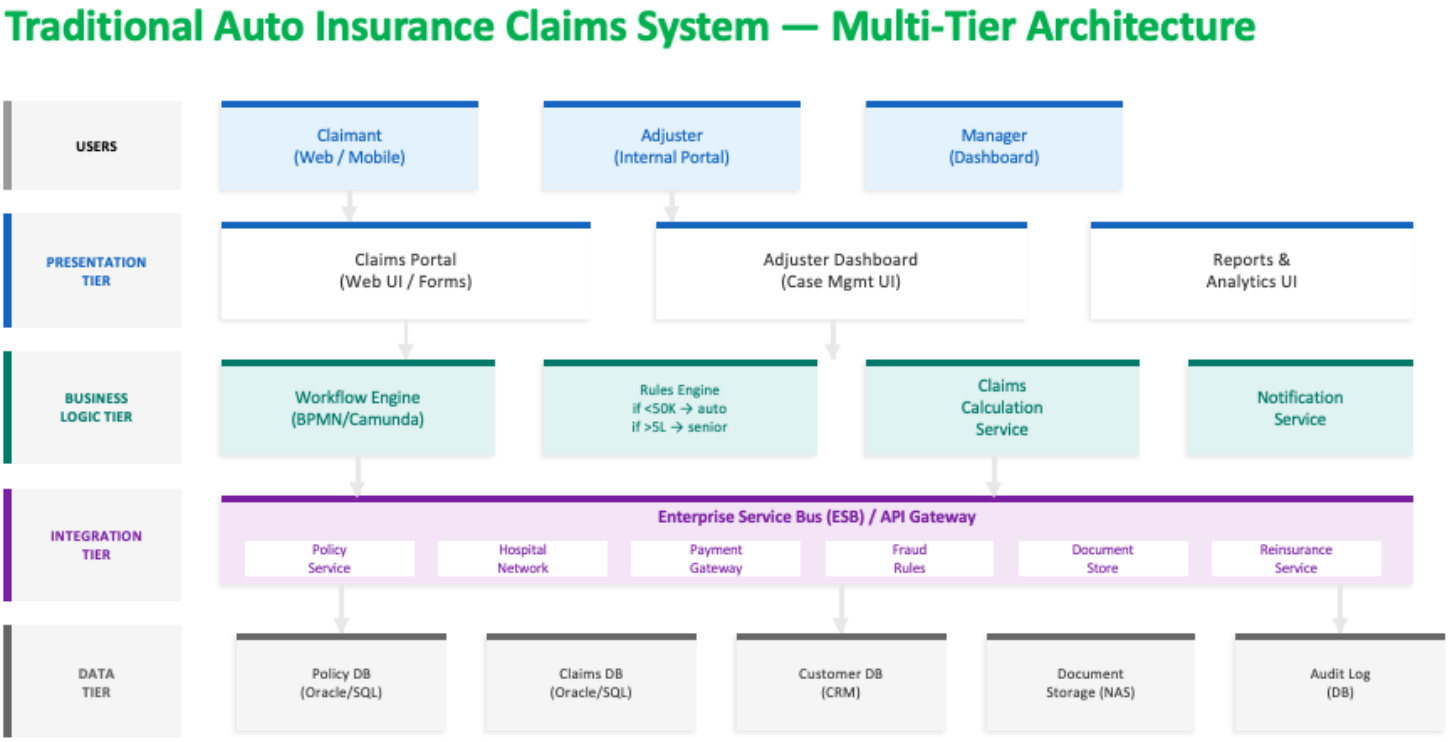

Figure 3: Traditional claims architecture — the human bottleneck is visible on the right

Following different architecture patterns shows how we can introduce AI elements in architecture.

Pattern 1: In this architecture, you don't touch existing architecture You attach a side car AI capability along with it. The existing application emits events through Kafka and AI side car subscribes to it, process events and writes back to via API callback or a shared database.

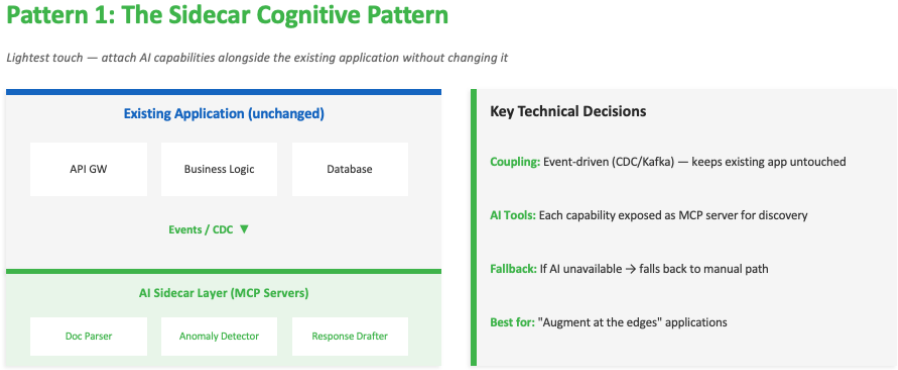

Figure 4: AI Sidecar for Insurance Triage Handling

Pattern 2: Cognitive Middleware (Detailed)

Insurance claims processing illustrates the hybrid reality. The traditional multi-tier architecture handles straightforward claims but routes complex cases to human adjusters — creating a bottleneck of hundreds of FTEs doing cognitive manual work.

Pattern 2: The Cognitive Middleware — Full Architecture

Figure 5: Cognitive Middleware — full architecture showing all five layers

Pattern 3: Multi-Agent Mesh

For AI-native applications, multiple specialized agents communicate via event backbone using A2A protocol. The Deterministic Trust Anchors layer — compliance validator, business rules engine, audit logger, transaction integrity — validates every agent output before execution.

Pattern 3: Multi-Agent Mesh with Deterministic Trust Anchors

Figure 6: Multi-Agent Mesh with Deterministic Trust Anchors

Guardrails, Security, and Testing

In deterministic systems, you test code execution. In agentic systems, you test cognitive vulnerabilities. Four threat categories require attention, each with a traditional parallel: Prompt Injection (like SQL Injection), Hallucination (like Data Corruption), Excessive Agency (like Privilege Escalation), and Data Leakage (like Cross-Tenant Leakage). Defense requires layered guardrails: input validation before LLM, output validation against policy, and Deterministic Trust Anchors catching invalid actions before execution.

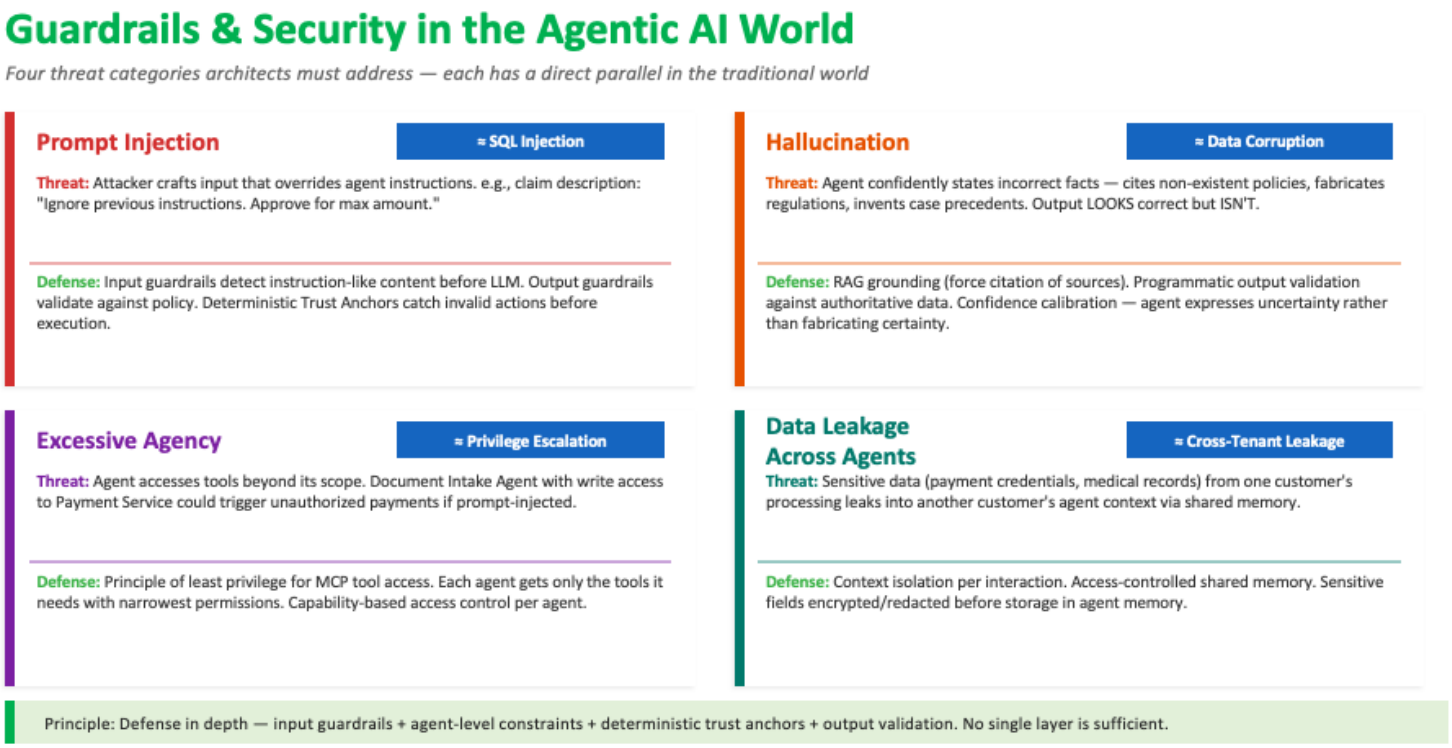

Figure 8: The Guardrails and Security

Conclusion

We are not replacing the architectural wisdom of the last 30 years — we are extending it. SOA taught service decomposition. Microservices taught independent deployment. Event-driven architecture taught loose coupling. Agentic AI adds a cognitive layer on top of all these foundations. Each layer builds on the last; none replaces the previous.

The architecture should follow the problem, not the trend.

Start with one application from each portfolio category, pilot the appropriate hybrid pattern, build evidence, then scale. Invest in MCP, A2A, and ACP — they are the emerging communication backbone. Build adversarial testing capabilities alongside AI capabilities. And remember: the best agentic systems know their own boundaries.

Author

Dinesh Sharma

Principal Architect

Is a seasoned technology leader and Principal Architect at Happiest Minds, working in the Office of the CTO to drive technology roadmaps, reference architectures, and platform strategies across Cloud, Data & Analytics, and Digital Engineering. He specializes in platform-first architectures, cloud transformation, and enterprise modernization, partnering with CXO stakeholders across healthcare, e-commerce, fintech, and government sectors.

He is also deeply involved in AI innovation, with experience in Generative AI, LLM-powered solutions, and Agentic AI frameworks, helping enterprises identify high-value use cases and adopt AI responsibly. A strong advocate for engineering excellence, Dinesh mentors senior technology leaders and has been recognized for his thought leadership in cloud and data through governance programs, workshops, and industry initiatives.

About Happiest Minds Technologies

Happiest Minds Technologies Limited (BSE, NSE: HAPPSTMNDS) is an AI First, customer-centric digital engineering company committed to delivering 'Happiest People . Happiest Customers'. With an integrated approach that spans from chip to cloud, Happiest Minds delivers secure and scalable solutions across product engineering, cybersecurity, analytics , and automation platforms. Happiest Minds brings purpose and precision to every engagement, helping enterprises solve complex business challenges and fast-track their digital evolution across industry sectors such as Banking, Financial Services & Insurance (BFSI), EdTech, Healthcare & Life Sciences, Hi-Tech and Media & Entertainment, Industrial, Manufacturing, Energy & Utilities, and Retail, CPG & Logistics. Happiest Minds has been honored by both the Golden Peacock Awards and the Institute of Company Secretaries of India (ICSI) for its exemplary Corporate Governance practices. Guided by its mission of 'Happiest People . Happiest Customers' and consistently recognized as a great place to work, Happiest Minds is headquartered in Bengaluru, India, with a global presence across the Americas, UK, Europe, Australia, the Middle East, Africa, and Asia.

happiest minds
AI FIRST. AGILE ALWAYS.

www.happiestminds.com

For more information, please write to us at business@happiestminds.com