Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| REVIEW ARTICLE

From SOA to Agents: What Three Turns of Platform Extensibility Teach Architects

Sandeep Gusain
Independent Researcher, USA
Corresponding Author: Sandeep Gusain, **E-mail**: connecttosandeepg@gmail.com

| ABSTRACT

Enterprise platforms have gone through three distinct extensibility models over the last two decades: service-oriented architectures built on SOAP and WSDL, RESTful microservices built on OpenAPI contracts, and, most recently, agentic systems built on natural-language intent and tool-invocation schemas. Each transition has been described in its own moment as an unprecedented break from what came before. This paper argues that framing is only partly correct. Drawing on a comparative qualitative analysis across all three eras, informed by first-hand architectural experience with SOAP/WCF integration, Azure-based microservice platforms, and an ongoing agentic extensibility initiative on the Microsoft Azure Portal, this paper identifies four principles that have remained constant across all three transitions: the interface a platform exposes is the product regardless of its notation; how a platform evolves that interface without breaking existing consumers determines the extensibility model's long-term viability; trust and permission boundaries must be treated as architecture rather than an added layer; and partner-facing ergonomics, not raw platform capability, decide whether an extensibility model is actually adopted. The analysis also identifies what is genuinely new in the agentic era: non-deterministic outputs that cannot be verified through conventional unit testing, capability contracts that must encode intent and reversibility rather than only parameters and return types, and trust boundaries that must now govern what an agent is permitted to decide, not only what data it can access. The paper concludes with a practical checklist for architects evaluating agentic platforms, and argues that engineers who have already navigated one or two of these transitions carry transferable judgment into the third, provided they distinguish genuinely new problems from old problems wearing new vocabulary.

| KEYWORDS

Agentic Extensibility, Service-Oriented Architecture, Microservices, API Governance, Platform Ergonomics

| ARTICLE INFORMATION

ACCEPTED: 15 August 2026 **PUBLISHED:** 10 September 2026 **DOI:** 10.32996/jcsts.2026.8.9.4

1. Introduction

A recurring pattern in enterprise software architecture is the arrival of a new integration paradigm accompanied by the belief that it invalidates the lessons of the paradigm it replaces. This happened when service-oriented architecture (SOA) displaced point-to-point integration in the early 2000s, and again when RESTful microservices displaced heavyweight SOAP services in the following decade. It is happening again now, as agentic artificial intelligence systems, capable of interpreting a stated intent and orchestrating actions across a platform's capabilities, are treated by many engineering organizations as a wholly new architectural problem rather than a third iteration of a familiar one.

This paper takes the position that the vocabulary changes with each transition, but a substantial part of the underlying architectural discipline does not. Teams encountering agentic extensibility for the first time are, in a meaningful sense, relearning lessons that platform architects who lived through the service-oriented and microservices transitions already paid for, sometimes at considerable cost. At the same time, treating the agentic transition as purely a repetition of prior patterns would be its own error:

non-determinism, intent-based contracts, and a redefinition of what a trust boundary must govern are genuinely new concerns without a direct precedent in the prior two eras.

This paper examines all three extensibility eras, service-oriented architecture, RESTful microservices, and agentic systems, through a comparative qualitative analysis grounded in first-hand architectural experience across all three: SOAP/WCF and TIBCO middleware integration for large-scale telecom platforms, RESTful microservice design for a global enterprise audit platform built on Microsoft Azure, and, currently, the design of the agentic extensibility model that lets partners surface Azure Copilot capabilities within their own customer experiences. The contribution is twofold: four durable architectural principles that recur across all three eras, and an honest accounting of what is genuinely different about the third.

2. Literature Review

2.1 Service-oriented architecture and the governance burden.

Service-oriented computing established the foundational premise that a platform's value lies in the contracts it exposes to consumers, rather than in the implementation details behind them (Papazoglou & Georgakopoulos, 2003). This was, at the time, a genuine advance over tightly coupled point-to-point integration. In practice, however, service contracts expressed through WSDL required substantial governance overhead: contract versioning, registry management, and cross-team coordination on shared interface definitions. Where this governance discipline was applied consistently, service-oriented architectures delivered on their reuse promise. Where it was treated as optional documentation rather than an architectural commitment, the same systems accumulated tight coupling and brittle dependencies despite their service-oriented label. This tension, between the promise of loose coupling and the discipline required to actually achieve it, is the first recurring theme this paper traces into the later eras.

2.2 Microservices, REST, and the shift of complexity to the network.

The move toward RESTful, resource-oriented service design reframed the same underlying problem in lighter-weight terms (Fielding, 2000). Fielding's architectural constraints, statelessness, a uniform interface, and cacheability, optimized for the scalability and independent evolvability of individual services, but did not eliminate the coordination problem that service-oriented architecture had already surfaced. It relocated it. Complexity that once lived inside a monolith's codebase moved into the network and into the organizational boundaries between teams owning independently deployable services. Recent empirical work on microservice API evolution, based on structured interviews with developers, architects, and managers across eleven companies, found that the central challenges are not technical in isolation but organizational: tight coupling reappears at the team level even when services are loosely coupled at the code level, and consumer reliance on outdated API versions is a persistent source of degradation (Lercher et al., 2024). The specific technology changed from WSDL to OpenAPI; the underlying problem, how a platform evolves a public contract without breaking the parties depending on it, did not.

2.3 Agentic systems and the tool-schema literature.

The most recent transition treats an agent's ability to discover and invoke a platform's capabilities as the central design problem, formalized through protocols such as the Model Context Protocol, which defines how an agent can enumerate available tools and invoke them through a standardized interface (Hou et al., 2025). Survey-level treatments of the shift from generative to agentic AI systems describe this as a genuine step change: agentic systems exhibit autonomous, multi-step reasoning and action in a way that earlier generative, single-turn systems did not, and this autonomy is what forces platforms to expose capabilities in a fundamentally different shape than a conventional API (Schneider, 2025). This literature is right that something new is happening. It is less attentive, however, to how much of the surrounding architectural discipline, contract stability, versioning discipline, and trust-boundary design, is not new at all, but a continuation of concerns already documented in the service-oriented and microservices literatures above.

2.4 Non-determinism and the limits of conventional testing.

A distinct and genuinely novel concern in the agentic literature is the difficulty of establishing confidence in a system whose output is not fixed for a given input. Recent work proposing an assessment framework for agentic AI systems argues that conventional software evaluation methods, built around deterministic, binary task-completion metrics, systematically overlook the behavioral uncertainty introduced by non-deterministic model outputs, and that a meaningful assessment must instead examine an agent's tool use, memory, and environment interaction as separate evaluation dimensions rather than a single pass/fail outcome (Parthasarathy et al., 2025). This has no direct analogue in either the SOA or microservices eras, where a given request against a given service version could be expected to produce a consistent, testable response. It is one of the two or three places in this paper's analysis where the honest answer is that the agentic era has introduced a problem prior extensibility models did not have to solve.

2.5 Documentation, partner ergonomics, and adoption.

A further recurring claim across all three eras is that partner-facing ergonomics, not raw platform capability, determine whether an extensibility model is actually adopted. This claim deserves scrutiny rather than repetition. A controlled study of software architecture documentation formats found no statistically significant relationship between documentation format, narrative versus structured, and a newcomer's ability to correctly answer architecture-understanding questions; what mattered far more was the newcomer's prior familiarity with the underlying source code (Ernst & Robillard, 2023). This is a useful corrective: platform ergonomics matter, but the mechanism is more specific than "better documentation wins." What this paper's later sections argue, consistent with this finding, is that ergonomics succeed when they reduce the gap between a partner's existing mental model and the platform's actual behavior, not simply when they are more thorough or more polished.

2.6 Positioning.

Taken together, this literature spans three eras of platform extensibility and a genuinely new concern, non-deterministic evaluation, introduced by the third. What is largely missing from this literature is a direct, first-hand comparative treatment connecting all three eras through a single architect's lived experience, distinguishing explicitly between what recurred and what is new. This paper offers that comparative treatment.

3. Methodology

This paper uses a qualitative, comparative case-analysis method across three architectural eras, rather than a quantitative or experimental design. This choice follows directly from the nature of the research question: identifying which architectural principles recur across three historically and technologically distinct extensibility paradigms is a question of pattern recognition across lived cases, not one that a controlled experiment or a statistical survey is well suited to answer. The three cases examined are drawn from the author's direct, first-hand architectural role in each: SOAP/WCF and TIBCO middleware integration for large-scale telecom web portals (T-Mobile.com and related properties), RESTful microservice design for a global enterprise audit platform (EY Canvas) built on Microsoft Azure, and the ongoing design of the agentic extensibility model for Azure Copilot within the Azure Portal, where partners consume Copilot capabilities inside their own customer-facing surfaces.

For each era, the analysis was structured around four fixed comparison dimensions, selected because each recurred as a source of both success and failure across the author's direct experience in all three eras: how the platform's interface is defined and exposed to consumers (the contract question); how that interface is changed over time without breaking existing consumers (the versioning question); how permission and trust boundaries are enforced (the trust-boundary question); and how easily a new partner or developer can begin building against the platform (the ergonomics question). Each dimension was evaluated qualitatively against the three cases, cross-referenced against the literature reviewed in Section 2, and used to separate genuinely recurring principles from claims that, on closer inspection, do not hold up consistently across all three eras. Section 4 presents the results of this comparison: four principles that held across all three eras, and four ways the agentic era departs from the pattern the first two eras established.

4. Results and Findings

4.1 What stayed the same: four durable principles.

Contracts are the product. Whether expressed as a WSDL document, an OpenAPI specification, or a tool-invocation schema, the interface a platform exposes is what a consumer actually experiences; the implementation behind it is invisible to them. Across all three of the author's cases, the systems that aged well were the ones where the team treated the public contract with the same rigor as the internal implementation, and the systems that aged poorly were the ones where the contract was allowed to drift informally while the implementation moved underneath it.

Versioning is destiny. How a platform evolves its public interface without breaking existing consumers is, in practice, the single largest determinant of an extensibility model's long-term viability. This was true of WSDL-versioned SOAP services, it remains true of OpenAPI-versioned microservices, where empirical research finds that consumer reliance on outdated versions and ineffective communication of breaking changes are the two most persistent failure modes across companies (Lercher et al., 2024), and it is, if anything, harder in the agentic era: a capability schema can drift not only in its formal signature but in the natural-language description an agent uses to reason about when to invoke it, a failure mode with no direct precedent in the first two eras.

Trust boundaries are architectural, not bolt-on. In every era examined, the security incidents and governance failures that mattered were the ones where authentication, authorization, or permission scoping had been treated as a layer added after the fact rather

than a property designed into the contract from the start. This was as true of SOAP-era security policies retrofitted onto existing WSDL contracts as it is of today's agent permission models.

Partner ergonomics decide adoption, but not for the reason usually assumed. It is tempting to state simply that better documentation wins. The evidence is more specific than that: a controlled study found that documentation format alone was not a statistically significant predictor of a newcomer's ability to understand a system's architecture, while prior familiarity with the underlying source code was (Ernst & Robillard, 2023). Applied to platform extensibility, this suggests that the platforms that actually got adopted, across all three eras examined here, were not necessarily the ones with the most exhaustive documentation, but the ones whose contracts most closely matched a new partner's existing mental model, so that less unlearning was required before the partner could be productive.

4.2 What genuinely changed with agents.

Four aspects of the agentic transition do not have a direct precedent in the prior two eras, and intellectual honesty requires stating them plainly rather than folding them into the durable-principles narrative above.

First, non-determinism. A SOAP service or a REST endpoint given a fixed input produces a fixed, testable output. An agentic capability, given the same stated intent, may reason its way to a different sequence of actions on a different invocation. Recent work proposing a four-pillar assessment framework, covering an agent's use of language models, memory, tools, and environment, argues explicitly that binary task-completion metrics inherited from conventional software testing fail to capture this behavioral uncertainty, and that a genuinely different evaluation approach is required (Parthasarathy et al., 2025).

Second, the contract now has to describe intent and context, not just parameters. A WSDL operation or a REST endpoint's parameters describe what data goes in and what comes out. A capability schema consumed by a reasoning agent has to describe what the capability is for, under what circumstances it should be invoked, and what it will not do, information that traditional interface definition languages were never designed to carry.

Third, evaluation replaces some of what testing used to guarantee. A team cannot unit-test its way to confidence in an agent's behavior the way it could unit-test a deterministic service method. This is not a minor tooling gap; it is a structural difference in how confidence gets established at all.

Fourth, the trust boundary now has to govern a decision, not just an access right. Where a SOA or microservices trust boundary answers the question "what data can this caller read or write," an agentic trust boundary must additionally answer "what is this agent allowed to decide on its own, versus what requires a human to confirm first," a question the earlier two eras never had to formalize because every action originated from a deliberate human click.

5. Conclusion

The central argument of this paper is not that agentic extensibility is simply SOA or microservices wearing new vocabulary. Both framings, that agentic systems are entirely unprecedented and that they are simply history repeating, are incomplete. Four architectural principles, contract discipline, versioning rigor, architected trust boundaries, and ergonomics grounded in matching a partner's existing mental model, recur across all three eras examined here and remain the load-bearing concerns of any extensibility model, agentic or otherwise. At the same time, non-determinism, intent-and-context-bearing contracts, evaluation in place of testing, and a trust boundary that now governs agent decisions rather than only data access, are genuinely new, and treating them as old problems in disguise would be its own mistake.

Table 1 offers architects a practical starting checklist, built directly from where this paper's three cases succeeded or struggled, for evaluating an agentic extensibility initiative against both the durable principles and the genuinely new concerns identified above. The broader claim is a modest one: engineers who have navigated one or two of these transitions already are not starting from zero when they encounter the third. Their advantage lies in knowing which of their prior scars are still relevant, and in being honest about the small number that are not.

Table 1

A Practical Checklist for Architects Entering the Agentic Era

Question	**Why It Matters**
What's your capability versioning story? Can a partner keep working against an older capability description while you evolve the newer one?	Versioning failures are the most common cause of partner breakage across all three eras (Lercher et al., 2024).
How does a partner test against your agent deterministically, given that the same input may not produce the same output twice?	Conventional integration testing assumes determinism; agentic systems require evaluation harnesses instead (Parthasarathy et al., 2025).
Where is the human-in-the-loop boundary, and is it enforced architecturally or only by convention?	Trust boundaries left as convention rather than architecture are where every era's security incidents originated.
Does your capability contract describe intent and reversibility, or only a function signature?	A schema that only describes parameters gives an agent no way to reason about what an action actually does.
What happens when a partner's integration was built against a capability description that has since drifted?	Prompt and capability drift is a new failure mode with no Era 1/Era 2 direct analogue.
Who owns the decision to expose a new capability to agents, and what review gate exists before it ships?	Governance overhead was SOA's most common complaint; skipping it in the agentic era reintroduces the same risk under a new name.
If a partner's documentation is thin, how long does it take a new developer to get a working integration?	Prior familiarity with the underlying system, not documentation format alone, is what predicts successful onboarding (Ernst & Robillard, 2023).

Statements and Declarations

Funding: This research received no external funding.

Conflict of Interest: The author declares no conflict of interest.

Acknowledgments: The author used Claude (Anthropic, claude-sonnet model) to assist with manuscript drafting and language refinement. The author takes full responsibility for the accuracy, originality, and integrity of all content presented in this manuscript, and confirms that all analysis and conclusions reflect independent professional judgment and verified sources.

References

[1]. Ernst, N. A., & Robillard, M. P. (2023). A study of documentation for software architecture. Empirical Software Engineering. https://doi.org/10.48550/arXiv.2305.17286

[2]. Fielding, R. (2000). Architectural styles and the design of network-based software architectures [Doctoral dissertation, University of California, Irvine].

[3]. Hou, X., Zhao, Y., Wang, S., & Wang, H. (2025). Model Context Protocol (MCP): Landscape, security threats, and future research directions. arXiv. https://doi.org/10.48550/arXiv.2503.23278

[4]. Lercher, A., Glock, J., Macho, C., & Pinzger, M. (2024). Microservice API evolution in practice: A study on strategies and challenges. Journal of Systems and Software. https://doi.org/10.1016/j.jss.2024.112110

[5]. Papazoglou, M. P., & Georgakopoulos, D. (2003). Service-oriented computing. Communications of the ACM, 46(10), 25–28. https://doi.org/10.1145/944217.944233

[6]. Parthasarathy, K., Akshathala, S., Adnan, B., Ramesh, M., Vaidhyanathan, K., & Muhammed, B. (2025). Beyond task completion: An assessment framework for evaluating agentic AI systems. arXiv. https://doi.org/10.48550/arXiv.2512.12791

[7]. Schneider, J. (2025). Generative to agentic AI: Survey, conceptualization, and challenges. arXiv. https://doi.org/10.48550/arXiv.2504.18875