Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

APIs as Digital Supply Chains: An Architectural Framework for Enterprise Data Logistics

Dreema Patel
Adobe, USA

ARTICLE INFO

Received: 08 April 2026

Accepted: 11 April 2026

ABSTRACT

APIs are an essential mechanism for orchestrating flows of information across systems, organizations and countries, but although considerations for well-formed components are widely established, an architecture framework based on the concept of 'API-as-supply-chain' is still very much in its infancy. Therefore, practitioners and researchers are currently unable to adapt the theory and applications of successful SCM constructs such as resilience, traceability, governance, and throughput optimization to the API context. This article tries to address this gap by proposing a five-layer API infrastructure architecture that reflects the SCM logistics functions of sourcing & production, processing & transformation, distribution & delivery, experience aggregation, and governance & compliance. The framework was developed according to a design science research methodology, and was validated through a formal expert review through the Analytic Hierarchy Process, with a consistency ratio of 0.044 by 7 domain experts. The model applies established integration patterns and supply chain management theory. Business ecosystems and monetization are horizontal concerns. The paper contributes to the research of API ecosystems as a value logistics system, and provides systematic guidance to practitioners on how to design resilient, governable and scalable digital infrastructures. Future areas of research include self-optimizing API ecosystems based on artificial intelligence (AI) and composable enterprise architecture.

Keywords: Api Architecture, Digital Supply Chain, Enterprise Integration, Platform Ecosystems, Design Science Research

1. Introduction

1.1 Background and Motivation

Modern enterprises are part of ever-expanding ecosystems of suppliers, customers, and partners. Data, services, and compute resources are continuously exchanged at the corporate boundaries of both organizations through APIs. APIs are the primary digital commerce platforms for exchanging trusted data between payment systems, identity services, document workflows, regulatory reporting, and AI inference endpoints. From a business value perspective, API and a surrounding ecosystem (infrastructure & tools) are assessed beyond technology boundaries in terms of how accessible the interface is to a company's partners, developers, regulators and customers rather than feature completeness of the product. Organizations interested in being API-first can achieve agility to respond to a volatile marketplace (Adewusi et al., 2021). Architectures such as microservices, GraphQL and RESTful APIs are focused on the concepts of modularization, rapid development and customizability for business-to-business and business-to-consumer applications.

Likewise, physical supply chains have extraction, production, storage, customs and trade enforcement, logistics and retailing layers, while digital supply chains use digital data generation, transformation and enhancement, routing and access control, aggregation and packaging, compliance and monitoring, and consumption and application or partner product integration layers instead. Similar to physical

supply chains, the extent of this downstream effect depends on latency, reliability, quality and governance. Digital transformation has impacted both processes and enterprise architecture across industries. In regulated industries such as finance, healthcare, logistics and telecommunications, agility, compliance with regulation, and capability to innovate contend with institutional inertia (Adewusi et al., 2021).

1.2 Problem Statement and Research Gap

API ecosystems and physical supply chains share similar structures, yet the former has been analyzed separately from the latter, with API management research focused on technical issues like protocols, security and performance. However, in the discipline of supply chain management, design and operational issues have not received similar attention, and supply chain research has not yet widely translated physical logistics concepts to digital logistics (technologies). Although theories in sustainable supply chain management and digital transformation can help explain the integration of digital technologies in logistics operations, little research has been conducted on API infrastructure following these approaches (Stroumpoulis & Kopanaki, 2022). The conceptual gaps that exist may also obstruct organizations' attempts to adopt established resilience engineering theories in their API ecosystems.

Enterprise organizations may have hundreds to thousands of internal microservices, partner-facing public APIs, and ecosystem solutions, and in multi-stakeholder enterprise digital ecosystems, regulatory compliance, data privacy, and interoperability challenges must be addressed to ensure trusted, secure, and scalable interoperation and integration between heterogeneous platforms (Adewusi et al., 2021). Just as a port closure can impact global supply chains, without architectural guards in place, closely interrelated core identity services such as authentication, transaction authorization, reporting and user notifications can impact each other through cascading failures.

1.3 Research Objectives and Contribution

To cover this research gap, a multi-layered architectural modeling framework is introduced in the paper. In detail, the goal of the paper is to (1) map the components of an API infrastructure to the layers of goods supply chains, (2) synthesize existing integration patterns with supply chain management theories into an architectural framework, and (3) validate the architectural framework through structured expert evaluation.

Theoretically, it uses the principles of the supply chain and value network theory to model an API value logistics system within digital infrastructures. Practically, it provides guidelines for enterprise architects to design a rapidly evolving, well-governed, and scalable API ecosystem. The paper is organized as follows: Section 2 discusses related work in API architectures, supply chain management and platform economics. Section 3 discusses the design science research. Architecture is covered in Section 4, Governance, compliance and security in Section 5. Section 6 covers business ecosystem and monetization perspectives as cross-cutting concerns. Section 7 discusses results of structured expert assessment. Section 8 discusses implications and limitations of the findings. Section 9 provides a conclusion and future research directions.

2. Related Work

2.1 API architecture and integration with enterprises

API architecture patterns have been researched in the areas of protocol design, security, and performance. In enterprise integration patterns, patterns such as Facade, Adapter, Circuit Breaker,

Research Article

and Event-Driven Architecture have been proposed for building scalable integration of multiple enterprise resource planning (ERP) systems (Ashok, 2023). Microservices architecture breaks a software application into small services that run independently and communicate using application programming interfaces (APIs) to achieve decentralization and shorter release cycles (Dragoni et al., 2017). Distributed frameworks for building cloud-native microservices show how stateful services with strong consistency can be provided in distributed microservices environments (Kakivaya et al., 2018). Network-centric distributed tracing models that do not require code instrumentation can support modern observability systems to track the execution of a request across multiple services (Shen et al., 2023). However, observability and monitoring complexity and volatility was described as the most pressing challenge in a qualitative interview study with 28 software experts. It has also been described as a prerequisite to service stability and the development of client applications (Niedermaier et al., 2019).

2.2 Supply Chain Management Theory

Supply chain management research has attempted to separate logistics into layers, namely sourcing, production, transformation, distribution, and delivery. Research into sustainable supply chain management and into digital transformation studied the role of digital technology in logistics (Stroumpoulis & Kopanaki, 2022). Similar approaches for the synthesis of the literature on supply chain social sustainability provide methodological guidance for developing systematic frameworks (Yawar & Seuring 2015). An integrative review of supply chain resilience capabilities (flexibility, velocity, visibility, collaboration) identifies a contribution to concepts related to supply chain digital infrastructure (Suthumdilok et al. in press). Architectural description based on software architecture research is foundational to understanding the structure and relationships of a system (Garlan, 1995).

2.3 Platform Economics And Ecosystem Theory

The research on MSPs generates perception into how value is created in platform ecosystems through network effects and complementor interactions, enabling for example the development of the concept of API marketplaces (Abdelkafi et al., 2019). Therefore, ecosystems could be referred to as a collection of firms that manage unique or supermodular complementarities designed for co-creating value (Jacobides et al., 2018). Ecosystems should have multilateral non-generic complementarities, homogeneous inter-role linkages, and decentralized decision-making. There is variation across corporate ecosystems in the interdependence of roles, the strength of the complementarities between the roles, and the non-genericity of the complementarities. These describe the conditions under which and the reasons why firms coalesce to form an ecosystem (Jacobides et al., 2018).

3. Research Methodology

3.1 Design Science Research Approach

The design science research methodology has been selected because it is the most appropriate methodology for constructing and evaluating frameworks, architectures, and models to resolve organizational issues. Unlike research that is focused on the understanding of reality, design science research is to create new artifacts that improve human and organizational capabilities. The DSR process model, described as a nominal process model, consists of six activities: problem identification and motivation, solution objectives, design and development, demonstration, evaluation, and communication (Peffers et al., 2020).

The framework includes principles from information systems design science: design as artifact, problem relevance, design evaluation, research contributions, research rigor, design as a search process, and research communication. It also addresses the fragmentation of API ecosystem design. The systematic guideline synthesis and validation process helps transfer supply chain frameworks to the evaluation of digital infrastructures. Furthermore, the guideline-based frameworks for the evaluation of a decision support system also serve the evaluation of DS research (Arnott & Pervan, 2012) and are hence applicable to the evaluation of information systems artifacts.

3.2 Framework Development Process

The architecture framework was developed in three phases. In the first phase, I studied the architecture of APIs in industry specifications and research. API architecture patterns were initially identified from (i) industry specifications/reference architectures from cloud service providers and API management vendors, (ii) peer-reviewed research literature from the Scopus and the Web of Science databases using the keywords: "API architecture", "API gateway patterns", "microservices integration", "enterprise integration patterns" from 2015 to 2024 and (iii) documentation from organizations with mature API programs. Supply chain management layer models were identified from searches of the keywords: "supply chain management framework", "logistics layer model", and "supply chain functional decomposition". Out of the 127 identified literature, we selected 43 papers to identify the concepts that apply to the framework. We stress papers on resilience, observability, and security.

In the second phase, functional maps of the provided API supply chains were created, using the functional layers from well-known supply chain management models for decomposing logistics functions (sourcing, production, transformation, distribution and delivery layers). The individual API components were then mapped to these layers. The five-layer framework has been further refined for terminology, completeness and compatibility through iterations in the third phase, resulting in a practically applicable and theoretically consistent framework.

3.3 Validation Approach

Expert evaluation (EE) is a commonly used method to evaluate design artifacts before they are fully realized. This is especially true for architectural frameworks because empirical validation may not be cost- or time-efficient. Design science methodological literature describes approaches to evaluation, including observational, analytical, experimental, testing, and the descriptive (Delport et al., 2024).

A panel of experts in enterprise API architecture, platform engineering and supply chain management evaluated the framework for completeness, consistency, clarity, applicability and novelty. AHP was used to quantify the differences in importance of parts of the framework to one another through expert opinion to grade and rank criteria based on professional judgment. Results are presented in section 7.

4. Architectural Framework

4.1 Framework Overview and Layer Structure

This architecture shows how API infrastructure can be organized using five layers. This follows the supply chain logistics analogy through raw material to system-of-record APIs and core data APIs; manufacturing to orchestration, workflow engines, and event brokers; distribution to gateway routing, API gateways, and load balancers; last mile to experience aggregation, backend-for-frontend applications, and GraphQL federations; and customs to governance, policy enforcement, and audit

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

systems. The business ecosystem and monetization are cross-cutting concerns, as they apply to all layers, and are not shown as a separate sixth layer (see Section 6).

The layer concept represents logical boundaries defined by functional decomposition. Components may be implemented on different infrastructure layers, but logically the components are separated by functional boundaries. This allows each layer to evolve independently of the system.

Supply Chain Function	API Architectural Layer	Primary Components
Raw Material Sourcing	System-of-Record APIs	Core data services, domain APIs, data products
Manufacturing & Assembly	Orchestration Layer	Workflow engines, event brokers, composition services
Distribution & Logistics	Gateway & Routing	API gateways, load balancers, traffic management
Last-Mile Delivery	Experience Aggregation	Backend-for-frontend services, GraphQL federation
Customs & Compliance	Governance & Security	Policy enforcement, observability, audit systems

Table 1: Mapping of **Supply** Chain Functions to API Architectural Layers [Garlan (1995)]

Note: Table 1 synthesizes mappings derived from supply chain functional decomposition frameworks (Yawar & Seuring, 2015; Stroumpoulis & Kopanaki, 2022), API architecture patterns (Dragoni et al., 2017; Ashok, 2023), platform infrastructure research (Kakivaya et al., 2018), and software architecture theory (Garlan, 1995).

4.2 Sourcing and Production Layer

At the base of all digital supply chains are system-of-record APIs that expose fundamental enterprise data resources, such as customer identity, product catalogs, pricing, compliance and audit information, financial ledgers, and machine statistics. At this level architectural stability with evolutionary potential applies. Core systems are often early monoliths with tightly coupled database schemas that may be wrapped with standard RESTful or GraphQL APIs to create modular production nodes.

To avoid cascading failures, system-of-record APIs must support versioning, backward compatibility, and strong consistency guarantees in addition to encapsulation. Distributed microservices deployments are generally implemented using distributed platforms for building cloud-native microservices. These platforms leverage the distributed infrastructure to provide strong consistency stateful services (Kakivaya et al., 2018) for stateful workloads. Many organizations adopt a layered approach to deployment that aligns with domain-driven design and data-as-product principles, assigning domain teams to specific bounded contexts with well-defined contracts.

To avoid compatibility issues, schemas may be stored in schema registries, API catalogs, or contract testing pipelines to prevent the introduction of incompatibilities. These approaches are similar to supplier certification programs used in global logistics to guarantee interoperable communication between independently evolving systems. Latency and durability modeling are also part of this layer. High throughput bursts, concurrency, and failure recovery are some of the challenges facing

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

production APIs. Read replicas, caching tiers, circuit-breakers and bulkheading patterns are used to prevent cascading failures.

4.3 Processing and Transformation Layer

Intermediary services, or orchestration APIs, handle data validation, cleansing and aggregation by tying in to upstream data repositories to return data in the format required by the calling service. Digital commerce checkout requests can, for example, involve identity checks, fraud scores, pricing, tax, inventory, shipping charges, and payment authorization using multiple production application programming interfaces (APIs) with orchestration responsible for sequencing, retry logic, compensation, and response normalization.

Microservices is a way of developing software where an application is divided into small independent microservices, interlinked through loose coupling, in order to isolate services and accelerate release cycles. Programming as an assembly line involves transforming raw materials through processes and quality assessment into finished products (Dragoni et al., 2017). Current orchestration architectures largely rely on event-driven architectures with message brokers and streaming platforms decoupling producers and consumers to allow for asynchronous flows and loosely coupled services.

The orchestration layer implements resiliency features such as timeout and retry policies, idempotency safeguards and fallback policies, which allow a request to be carried through pipelines even if some upstream service is not operational. Observability is another important feature that the orchestration layer provides, for example using distributed tracing to visualize how a request propagates across services, helping identify latency bottlenecks. Likewise, the throughput of APIs can often be modeled in terms of minimizing latency, for example, through the parallelization of individual API calls wherever possible.

4.4 Distribution and Delivery Layer

API gateways are the international ports of call in a digital supply chain. They are distributed systems that control upstream and downstream traffic, authentication and rate limiting, and request routing to upstream systems. Port facilities check, document, and confirm shipments prior to clearing customs. API gateways verify tokens, read message payloads, implement quotas, and check compliance.

Modern observability solutions that do not require instrumenting code typically rely on network-centric distributed tracing models to correlate a request's progress across multiple services (Shen et al., 2023). Gateways may also support protocol translations between REST, GraphQL, gRPC, and legacy SOAP. Enterprise use of hybrid and multi-cloud deployments encourages gateways that make use of smart routing for optimized latency, costs and regulation.

Enterprise Integration Patterns can help solve problems that come with APIs that interface with more than one ERP. Examples are the Facade, Adapter, Circuit Breaker and Event-Driven Architecture patterns (Ashok, 2023). Gateways should be able to cope with burst traffic without becoming bottlenecks. While horizontal scalability, queuing, caching and adaptive rate limiting help achieve a constant throughput and prevent overload, the last step of distribution is context-specific and might be implemented using Backend-for-Frontend services or GraphQL federation rather than forwarding calls from the user to multiple microservices directly.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

5. Governance, Compliance, and Security

5.1 Policy Enforcement and Zero-Trust Architecture

Security and compliance regulate an API supply chain as customs and border protection. Every request in a zero-trust model is authenticated, authorized, and inspected, regardless of its source or network location. Authentication and authorization standards, including OAuth, mutual TLS, API keys, and token exchange patterns, secure data exchanged between organizations, and fine-grained authorization rules control visibility and modification permissions.

Regulated industries have requirements for data residency, audit logging, retention, consent enforcement, and encryption. These controls can be integrated inline in traffic pipelines via an API management platform, with governance at scale, rather than enforced ad-hoc at the individual service level. This reduces operator error and configuration drift, allowing regulatory compliance across a distributed network in the same way as customs inspection protocols in physical logistics.

Governance controls are also possible throughout the layers. In production layers, schema validation and contract tests can be applied. In the orchestrator layer, role-based features are available as orchestrator roles for compensating transactions and policy enforcement for workflow engines. In gateway policy engines, role-based features are available like authentication, rate limiting, and routing . In aggregation layers, there are features like response filtering and query authorizations for field-level access controls. Centralized observability systems include audit logging and anomaly detection.

Framework Layer	Primary Governance Controls	Enforcement Mechanism
Production	Schema validation, contract testing	CI/CD pipeline gates
Orchestration	Transaction integrity, compensation	Workflow engine policies
Distribution	Authentication, rate limiting, routing	Gateway policy engine
Aggregation	Response filtering, field-level access	Query authorization
Observability	Audit logging, anomaly detection	Centralized monitoring

Table 2: Governance Control Distribution Across Framework Layers

5.2 Observability and Traceability

Real-time visibility is necessary for digital supply chains. Observability platforms often provide distributed tracing, logging, and metrics about API calls to garner a better understanding about data and requests flowing through a software system. According to a qualitative industry interview study of 28 software professionals, observability and monitoring complexity and volatility were identified as the most pressing challenge by interviewees, as well as a prerequisite for service stability and the development of client applications (Niedermaier et al., 2019).

According to the interviewees, microservices, DevOps and cloud development and operation decoupled observability tools from each other and customer or business views because they maximize independence and specialization (Niedermaier et al., 2019). The nine identified challenges are increasing dynamism and complexity, heterogeneity, company culture and mindset, lack of central point of view, flood of data, dependency on experts, lack of experience and resources, unclear non-functional requirements, and reactive implementation (Niedermaier et al., 2019).

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

End-to-end traces are critical for financial services, healthcare and mission-critical compliance scenarios, where an audit or investigation involves reconstructing a transaction or flow. End-to-end traces enable incident response, SLA performance verification, and forensic auditing across service boundaries and in heterogeneous service provider environments. Similar to physical supply chains, where tracking numbers, manifests, and scanning events are used to establish product provenance and give a history of product location and status, in distributed observability architectures production telemetry data flows from production APIs through orchestration and distribution layers to centralized observability platforms with trace correlation identifiers propagating across service boundaries to reconstruct end-to-end transactions.

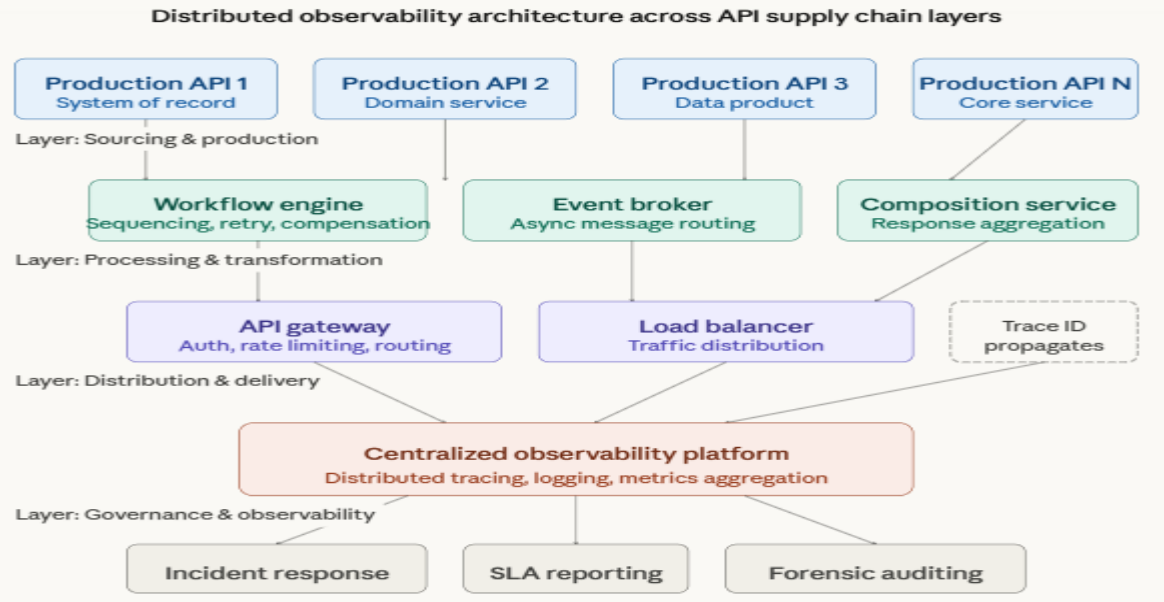

Figure 1: Distributed Observability Architecture Across API Supply Chain Layers [Niedermaier et al. (2019), Shen et al. (2023)].

The most common architecture shown in figure 1 is to have production APIs send their telemetry data through orchestration and distribution layers to be stored in a central observability platform. The various layers of the architecture are the sourcing layer (Production API 1 - System of record, Production API 2 - Domain service, Production API 3 - Data product, Production API N - Core service). (2) Processing & Transformation layer: Workflow engine (Sequencing, retry, compensation), Event broker (Async message routing), Composition service (Response aggregation). (3) Distribution & Delivery layer: API gateway (Auth, rate limiting, routing), Load balancer (Traffic distribution). (4) Governance & Observability layer: A centralized observability platform (Distributed tracing, logging, metrics aggregation). The trace correlation identifiers propagate across service boundaries as downward arrows. The observability platform's output arrows are used for three operational functions: incident response, SLA reporting and forensic auditing.

6. Business Ecosystems and Monetization

Business ecosystem and monetization are cross-cutting concerns throughout all five layers of the architectural model, that is, they apply to each architectural layer. They describe how business and financial value flows between the various participants in the API supply chain, and how that value is captured and allocated within the organization.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

6.1 Platform Extension and Partner Integration

APIs extend supply chains across enterprise boundaries into partner ecosystems. Payment providers, identity providers, logistics providers, and Software-as-a-Service suppliers expose their capabilities and products as consumable APIs to enable embedded finance, marketplace onboarding, third-party services, and composable business processes respectively. In this sense, ecosystems are sets of firms that have to manage specific co-specialized or supermodular complementarities in order to co-generate value (Jacobides et al., 2018).

Ecosystem benefits consist in organizing independent and interdependent firms to exploit complementarities in production and consumption without vertical integration through, e.g., standardized formal or informal agreements (Jacobides et al., 2018). In this sense, an ecosystem is thus conceptualized as a number of actors with multilateral heterogeneous non-generic complementarities not fully hierarchically controlled. Ecosystems require: (1) multilateral non-generic complementarities, (2) homogeneous linkages across each role, and (3) decentralized decision-making (Jacobides et al., 2018).

Further revenue models include charging per unit of usage, revenue sharing, and tiered subscriptions based on usage or value perception, while API marketplaces allow self-service onboarding. Value is created in these cases by connecting platform sponsors with complementors in order to deliver value to consumers (Abdelkafi et al., 2019). In the platformization of service providers, firms evolve to become either exporters of tradable digital capabilities or exporters of APIs (application programming interfaces).

6.2 Ecosystem Governance and Value Distribution

APIs and third-party use increasingly complicated issues of governance, while the ability to use, access, and monetize the API raised concerns about finding the balance between openness, governance, and control. These frameworks define property rights, liability, service levels and intellectual property rights for ecosystem operations.

Modularity addresses inter-firm coordination problems similar to, but not including, those facing independent but interdependent firms (Jacobides et al., 2018). Corporate ecosystems may differ in the kind and direction of interdependencies, in the strength of complementarity, and whether the complementarities are unique or supermodular. These ecosystem features help explain when and why firms rally around common objectives, and when they drift apart (Jacobides et al., 2018).

The ecosystem governance model maps governance dimensions to different implementation mechanisms for the internal APIs, partner APIs and public APIs. Such implementation mechanisms include role-based access control mechanisms for internal APIs, contracts for partner APIs and keys for public APIs. Service communities have rate limits for internal resource categories, partner tiers defined by service level agreements, and public usage caps. Versioning covers an internal deprecation phase, partner system migration, and a long-lived stability guarantee period for public interfaces. Common monetization strategies include chargeback systems, revenue sharing, and usage-based billing with pro-rated pricing for public access. Common support strategies include internal site reliability engineering, dedicated partner support, and public-facing self-service documentation.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Governance Dimension	Internal APIs	Partner APIs	Public APIs
Access Control	Role-based	Contract-based	Key-based
Rate Limiting	Resource quotas	SLA tiers	Usage caps
Versioning	Deprecation windows	Migration support	Long-term stability
Monetization	Cost allocation	Revenue share	Usage billing
Support Model	Internal SRE	Dedicated support	Self-service

Table 3: Ecosystem Governance Model for API Supply Chains

7. Validation Results

7.1 Expert Evaluation Methodology

Expert assessment techniques were used to validate the framework. To this end, a seven-member expert panel was formed, comprising PhD-educated experts in engineering and informatics who have years of experience assessing systems (Talero-Sarmiento et al., 2024). They were three enterprise API architects with experience in financial services and technology companies, two leaders of platform engineering in the cloud infrastructure industry, and two researchers in supply chain management (SCM) with in-depth knowledge of digital transformation. The participants had an average 14 years of experience (ranging from 8 to 22 years). This was a balanced representation of the technical, operational, and theoretical viewpoints.

Evaluators were not able to communicate with one another, nor were they aware of how other evaluators scored the proposal, or how important their evaluation was to selection.

The relative importance of different parts of the framework are quantified using the analytic hierarchy process (AHP) to grade and rank criteria based on expert opinion. The AHP method is considered to be natural, easy, reliable and stable (Talero-Sarmiento et al., 2024). The method requires defining the assessment goal, application context and a list of criteria. The experts must provide the pairwise comparison matrices for each criterion using a set of values on the Saaty scale ranging from 1 (equal importance) to 9 (extreme importance).

In practice, the five architectural layers were broken into eight assessable components. The Distribution & Delivery layer was collapsed into a single assessable component (Gateway Layer). The Sourcing & Production layer was evaluated as System-of-Record, the Processing & Transformation layer as Orchestration, while Governance & Compliance was treated as three components to evaluate: Governance & Observability, Security Policy Enforcement and Compliance Controls. Experience Aggregation was reviewed as a stand-alone business concern, as was Ecosystem Monetization, another cross-cutting business concern of the solution.

7.2 Quantitative Assessment Results

It led to AHP results with maximum eigenvalue 15.97, whose normalized weights are more than one scale across alternatives (Talero-Sarmiento et al., 2024). In this regard, a consistency ratio of less than or equal to 0.10 (or 10%) is a satisfactory result of user judgments in terms of AHP scale interpretation criteria. Since the consistency ratio (CR) of our AHP analysis was 0.044, much lower than its acceptable threshold of 0.1, the structure of the AHP analysis was consistent, and the expert

assessments were reliable (Talero-Sarmiento et al., 2024). Furthermore, the applicability of the framework was rated on a five-point Likert scale by all seven experts and considered as applicable or highly applicable to their organizational contexts.

In layers of the framework, the weight percentage is distributed among the components that define their importance level, where the maximum percentage is 18.96 and the lowest is 1.45.

Framework Component	Weight (%)	Relative Priority
Distribution & Delivery (Gateway Layer)	18.96	Highest
Sourcing & Production (System-of-Record)	18.06	High
Processing & Transformation (Orchestration)	9.26	Moderate-High
Ecosystem Monetization	7.75	Moderate
Governance & Observability	7.74	Moderate
Experience Aggregation	6.99	Moderate
Security Policy Enforcement	5.06	Lower
Compliance Controls	4.21	Lower

Table 4: Expert-Derived Weights for Framework Layer Components [Talero-Sarmiento et al. (2024)]

The most highly weighted layer was the distribution and delivery gateway layer with a weight of 18.96 percent, followed closely by the sourcing and production system-of-record layer with a weight of 18.06 percent. The processing and transformation orchestration layer had a moderate-high weight of 9.26 percent. The ecosystem monetization and governance and observability layers received the lowest weightings of 7.75 and 7.74 percent respectively, followed by experience aggregation (6.99 percent) and security policy enforcement and compliance controls (5.06 and 4.21 percent).

The experts agreed on the importance of the gateway and system-of-record layers. Their respective standard deviations were 1.2 and 1.4. The experts disagreed on the importance of compliance controls and security policy enforcement, which had higher standard deviations of 2.1 and 1.9, respectively, indicating that their importance depends on the context.

To evaluate the improvement this specialized pairwise comparison algorithm makes, we compare the number of comparisons in both cases. We show that while maintaining the number of evaluator comparisons, this approach reduces the number of pairwise comparisons from 105 to 30 with more consistent choices in 10000 cases. Of these cases, 858 had a consistency value lower than the threshold value of 0.1 (Talero-Sarmiento et al., 2024).

7.3 Qualitative Feedback Synthesis

Evaluations of the framework were categorized into three groups. With respect to framework utility, one evaluator stated that the supply chain analogy "provides a vocabulary that resonates with executive leadership" and "bridges the gap between technical architecture discussions and business strategy conversations". Alluded to practical applicability, "the layer model maps well to how we already organize platform teams" and "the governance model addresses a tension we struggle with daily - enabling developer velocity without losing control". In terms of completeness, two reviewers

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

suggested treating observability and security as cross-cutting concerns instead of layers; this is a possible direction for future work.

This could help to address the issues of bias from different scales and the burden of making multiple comparisons (Talero-Sarmiento et al., 2024). These results might support the argument for expert assessment to provide a complete understanding of architectural evaluation when compared to perception-based assessment (Talero-Sarmiento et al., 2024). For example, in this proposed framework approach, the rates of importance given by the evaluation leaders take precedence over the number of items, the scale, and the number of framework components, allowing the evaluations to prioritize specific aspects from the beginning (Talero-Sarmiento et al., 2024). Also, regarding assessments of design science research artifacts, choices for the evaluation approach such as expert evaluation are recommended according to the maturity of the artifact and research objectives (Cleven et al., 2009).

Expert Feedback Theme	Representative Comment	Framework Implication
Executive communication utility	"Supply chain analogy resonates with leadership"	Confirmed as practitioner benefit
Organizational alignment	"Layer model maps to how we organize platform teams"	Validates structural decomposition
Governance tension resolution	"Addresses developer agility vs. corporate control"	Validated governance model contribution
Cross-cutting concerns	"Observability and security span all layers"	Noted for future framework refinement

Table 5: Expert Feedback Themes and Framework Implications

8. Discussion

8.1 Research Objectives Revisited

Research objective 1 from Section 1.3, mapping components of API infrastructure onto functional layers within the supply chain, was addressed through the use of the five-layer framework in Section 4 and Table 1. We have achieved our second objective, of combining integration patterns with supply chain management theory, with the iterative development of our framework (section 3.2), which combines API architectural patterns with the logistics functional decomposition. We have also achieved our third objective, of validating our framework with a structured expert assessment procedure, with our AHP-based expert assessment (section 7), which achieved high consistency (CR = 0.044), and identified our gateway and system-of-record layers.

8.2 Theoretical Implications

It presents analogies between physical supply chain logistics ecosystems and digital API ecosystems to extend supply chain management theories within logistics, such as resilience engineering, throughput, and partner governance theory, to API ecosystem design. Furthermore, it provides an integrative review of the supply chain resilience capabilities within logistics ecosystems in terms of flexibility, velocity, visibility, and collaboration, as well as aspects of the supply chain resilience theory related to research on API ecosystems (Suthumdilok et al., in press).

The theory contributes to information systems theory by understanding APIs as logistics infrastructure for value, in addition to the existing understanding of APIs as integration infrastructures, and by adding a degree of specificity to theories of platforms in the platform economy . Studying multi-sided platforms has helped explainhow a platform ecosystem's value is created through network effects and the cooperation of complementors, which has helped inform research on API marketplaces (Abdelkafi et al., 2019).

8.3 Practical Implications

The model acts as a guide for practitioners who want to take a structured approach to the design of API ecosystems from a technical, operational or planned perspective. The layers can be adopted individually. As an example of microservices in practice research, some concerns and behaviors of engineers while working with distributed systems were identified and confirmed that the framework originally targeted real-world integration challenges (Viggiato et al., 2018).

The governance model also tackles one of the most common problems in enterprise API programs: the need to balance developer agility and freedom with corporate governance and organizational control. Governance rules are applied as part of the architecture rather than as an external constraint.

8.4 Limitations

One disadvantage of the framework is that it focuses only on architecture and does not specify any technology or implementation, which means that the organization can use any technology they desire. Further, validation of the framework in and of itself through expert evaluation is insufficient. Practical validation through its use and longitudinal studies could lend credence to the claim.

Third, the framework assumes that the organization is capable of adopting layered architectures, which may be difficult for smaller organizations or teams with meaningful technical debt. Fourth, this framework does not capture every possible threat scenario, and some threats (e.g. insider threats, compromised CI/CD pipelines, or advanced persistent threats) may require more controls than described in the governance layer.

Conclusion and Future Directions

Summary of Contributions

The article offers a layered reference model for APIs as digital supply chains, consisting of the layers sourcing and production, processing and transformation, distribution and delivery, experience aggregation, and governance and compliance. Business ecosystem and monetization aspects are addressed as cross-cutting issues. The framework applies the principles of supply chain management to digital infrastructure design.

The framework was created according to design science methodology and assessed by synthesis and expert review, where the inter-rater agreement coefficient was 0.044, indicating a high level of applicability of the framework. Members also report using the framework to ease communication between technical and business personnel.

Implications for Research and Practice

The framework builds on the theoretical foundations of supply chain management and digital infrastructure. APIs are regarded as logistics systems for value. Enterprise architects use the framework to design API ecosystems with strong governance and scalability within organizations. An organization implementing the framework can expect improvements in aligning API infrastructure

investment with business value delivered, phased delivery through a layered model and balanced innovation and time to market against regulatory and compliance requirements through governance.

Future Research Directions

Further optimizations could be done such as using AI to route traffic, automatically predict or forecast demand, and automatically scale compute and storage resources, route traffic around failures or congestion in the computational infrastructure, and automatically scale up or down the infrastructure. It is suggested that the framework offers some form of autonomous optimization, intended to help the framework stay relevant in next-generation architecture.

Second, data mesh and composable enterprise architectures decentralize ownership of the supply chain to domains with interoperable APIs for their products, services, and experiences. They are thus good candidates for further investment in federated governance and cross-domain compositions. Third, when generative AI and other advanced analytics are commoditized services exposed via API, companies will need data, workflows, and intelligence traversing any digital supply network. Finally, future longitudinal empirical studies needed to examine the effect of different framework adoptions on practice would strengthen the validity claim and elicit feedback on framework refinements.

References

[1] Abdelkafi, N., Raasch, C., Roth, A., & Srinivasan, R. (2019). Multi-sided platforms. Electronic Markets. https://link.springer.com/content/pdf/10.1007/s12525-019-00385-4.pdf

[2] Adewusi, B. A., Adekunle, B. I., Mustapha, S. D., & Uzoka, A. C. (2021). Advances in API-centric digital ecosystems for accelerating innovation across B2B and B2C product platforms. IRE Journals. https://www.researchgate.net/profile/Bolaji-Adekunle/publication/392470724.pdf

[3] Arnott, D., & Pervan, G. (2012). Design science in decision support systems research: An assessment using the Hevner, March, Park, and Ram guidelines. Journal of the Association for Information Systems, 13(11), 923-949. https://www.researchgate.net/profile/David-Arnott-2/publication/286316946

[4] Ashok P. P. K. (2023). Design patterns for scalable API integration in multi-platform ERP environments. Journal of Mathematical & Computer Applications. https://www.researchgate.net/profile/Paul-Praveen-Kumar-Ashok/publication/400602871_Open.pdf

[5] Cleven, A., Gubler, P., & Hüner, K. M. (2009). Design alternatives for the evaluation of design science research artifacts. Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology. https://doi.org/10.1145/1555619.1555645

[6] Delport, P. M. J., Gerber, M. C., & van der Merwe, A. (2024). Methodological guidelines for design science research. Procedia Computer Science. https://doi.org/10.1016/j.procs.2024.05.096

[7] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, today, and tomorrow. arXiv. https://arxiv.org/pdf/1606.04036

[8] Garlan D. (1995). Research directions in software architecture. ACM Computing Surveys, 27(2). https://doi.org/10.1145/210376.210388

[9] Jacobides, M. G., Cennamo, C., & Gawer, A. (2018). Towards a theory of ecosystems. Strategic Management Journal, 39(8), 2255-2276. https://sms.onlinelibrary.wiley.com/doi/pdf/10.1002/smj.2904

[10] Kakivaya, G., Xun, L., Hasha, R., Paber, S. B., Blank, B., Tantawi, A., ... & Theimer, M. (2018). Service Fabric: A distributed platform for building microservices in the cloud. Proceedings of the Thirteenth EuroSys Conference. https://doi.org/10.1145/3190508.3190546

[11] Niedermaier, S., Koetter, F., Freymann, A., & Wagner, S. (2019). On observability and monitoring of distributed systems—An industry interview study. arXiv. https://arxiv.org/pdf/1907.12240
[12] Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2020). The design science research process: A model for producing and presenting information systems research. arXiv. https://arxiv.org/pdf/2006.02763
[13] Shen, J., Wang, Y., Yang, J., Li, Y., Sun, Z., Yu, J., ... & Zhang, J. (2023). Network-centric distributed tracing with DeepFlow: Troubleshooting your microservices in zero code. Proceedings of the ACM SIGCOMM 2023 Conference. https://doi.org/10.1145/3603269.3604823
[14] Stroumpoulis, A., & Kopanaki, E. (2022). Theoretical perspectives on sustainable supply chain management and digital transformation: A literature review and a conceptual framework. Sustainability. https://www.mdpi.com/2071-1050/14/8/4862
[15] Suthumdilok, P., Suthiwartnarueput, K., & Pornchaiwiseskul, P. (2026). Towards a theory of supply chain resilience: An integrative review. Sustainability. https://www.mdpi.com/2071-1050/18/5/2497
[16] Talero-Sarmiento, L., Gonzalez-Capdevila, M., Granollers, A., Lamos-Diaz, H., & Pistili-Rodrigues, K. (2024). Towards a refined heuristic evaluation: Incorporating hierarchical analysis for weighted usability assessment. Big Data and Cognitive Computing. https://www.mdpi.com/2504-2289/8/6/69
[17] Viggiato, M., Terra, R., Rocha, H., Valente, M. T., & Figueiredo, E. (2018). Microservices in practice: A survey study. arXiv. https://arxiv.org/pdf/1808.04836
[18] Yawar, S. A., & Seuring, S. (2015). Management of social issues in supply chains: A literature review exploring social issues, actions and performance outcomes. Journal of Business Ethics. https://doi.org/10.1007/s10551-015-2719-9