International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Modernizing SAP Logistics through Real-Time Event-Driven API Architecture for Multi-Carrier Shipping Systems

Sudeepta Rana*

Senior Application Consultant, Tampa, USA
Email:sranaofficial2026@gmail.com

Abstract

Many SAP-centered logistics landscapes in ongoing S/4HANA transformation programs still retain fragmented carrier connectivity, delayed shipment visibility, and brittle point-to-point integrations inherited from earlier ERP environments. The proposed analytical model links architectural principles, integration layers, and governance decisions that support scalable shipping across heterogeneous carriers. The materials consist of ten recent academic sources on event-driven systems, microservice event management, digital freight platforms, production logistics, collaborative logistics, and digital transformation, complemented by a practitioner brief that frames SAP OTC, logistics execution, transportation management, and AMS realities. The analysis produces three outputs: a rationale for event-driven modernization, a reference architecture for SAP-centric shipping integration, and an implementation sequence for controlled migration. The model helps architects separate transactional authority in S/4HANA from carrier-facing execution, visibility, and exception services during phased migration.

Keywords: SAP S/4HANA; logistics modernization; event-driven architecture; API integration; multi-carrier shipping; transportation visibility; microservices; OTC architecture; shipping orchestration; enterprise integration.

Received: *5/1/2026*
Accepted: *7/1/2026*
Published: *7/11/2026*

---

* *Corresponding author*.

1. Introduction

Multi-carrier shipping has become one of the most fragile areas in ERP-centered logistics landscapes. Order creation, delivery processing, freight execution, tracking updates, exception handling, and proof-of-delivery feedback often span systems that were connected at different times, under different assumptions, and for different operational horizons. Formal system links remain in place, while daily shipment execution fragments across carriers, tracking tools, and exception channels. Data arrives late, carrier-specific interfaces accumulate, and business users compensate with manual tracking, spreadsheet reconciliation, and informal escalation chains.

SAP logistics modernization has to address that structural problem at the level of integration design, carrier connectivity, and shipment-state control. The analysis isolates the causes of failure in traditional integration patterns, defines the components of a SAP-centric event-driven API model, and sets out a migration logic for governance and operational control during S/4HANA programs.

The article draws together enterprise integration research, logistics digitalization studies, and shipping-visibility literature into a single SAP-oriented architectural model. The argument focuses on architectural composition, operational control, and migration sequence in SAP-centered shipping modernization.

2. Materials and Methods

2.1. Materials

The analytical corpus contains ten recent sources published between 2022 and 2025. They cover event-driven processing and distributed integration design [1; 6; 7], enterprise digital transformation and system flexibility [8], digital freight platforms and shipment visibility [3; 4], urban and collaborative logistics architectures [2; 9], and digital service architecture in production logistics [10]. One contemporary logistics operations study was added because it links real-time coordination with service modularity and decision support in execution-intensive environments [5]. Alongside the academic corpus, the analytical frame draws on practitioner material from senior SAP S/4HANA OTC consulting in regulated and high-volume distribution environments. That material covers OTC architecture, SAP–non-SAP carrier integration, cross-module dependencies across TM, FI, MM, WM, and GTS, conversion readiness for ECC-to-S/4HANA migration, controlled sandbox validation, and AMS stabilization. The source selection prioritizes direct relevance to real-time logistics integration, architectural design, and enterprise modernization, with preference for recent peer-reviewed publications and publisher-hosted records.

2.2. Methods

The analytical design uses comparative source analysis, conceptual synthesis, typologization of integration functions, and analytical generalization. These methods are aligned with the three research objectives. First, the sources are compared to isolate recurrent integration limits in distributed shipping environments. Second, their architectural propositions are synthesized into a reference model suited to SAP-centered logistics execution. Third, the identified design principles are translated into an implementation sequence focused on governability, migration control, and operational monitoring.

3. Results

Traditional integration patterns lose reliability in multi-carrier shipping when response time, state propagation, and exception handling remain tied to rigid point-to-point links or coarse batch cycles. Event-driven processing was developed for situations in which multiple applications need to react to operational state changes without waiting for a centralized polling routine [6]. Recent microservice research shows that organizations quickly encounter difficulties with event schema design, payload size, auditing, ordering, and troubleshooting when event flows are introduced without disciplined management practices [7]. Calvio and his colleagues argue that synchronous and asynchronous interaction should coexist because different operational dependencies require different communication semantics [1]. Wurm and his colleagues link ERP modernization to an explicit transformation blueprint [8]. In SAP shipping landscapes, this means that real-time architecture should be treated as a business design decision embedded in S/4HANA migration logic and governed at the level of process architecture, carrier connectivity, and execution control. In SAP programs with heterogeneous carrier landscapes, migration sequencing depends on early gap analysis between ECC-era interface logic and S/4HANA process design, followed by sandbox validation, controlled test environments, and release governance tied to business-critical shipment events.

A more logistics-specific comparison sharpens that conclusion. Road freight platform research identifies visibility, optimization, and analytics as recurring service families in digital forwarding environments [4]. Habjan shows that connected systems and electronic logistics marketplaces reduce waiting times, improve document handling, and strengthen cross-organizational coordination when real-time information exchange is integrated into execution routines [3]. Galkin and his colleagues place real-time analytics at the center of dynamic freight control because static management logic cannot absorb rapid operational variation [2]. Collaborative logistics research extends this idea to multi-party coordination and shows why loosely coupled architectures become attractive when actors hesitate to share data through a tightly integrated monolith [9]. Taken together, multi-carrier shipping breaks down when shipment-state awareness fragments across organizational boundaries. A modern SAP shipping design, therefore, needs a shared event backbone that can propagate shipment-state changes quickly while preserving partner autonomy and carrier heterogeneity.

The architectural composition of a SAP-centered event-driven shipping model rests on four requirements. Based on the architectural studies reviewed, four main requirements are identified. First, the integration layer must support asynchronous publication of business events so shipment states can be propagated immediately across dependent services [6]. Second, the integration layer needs explicit governance for event schemas, observability, replay procedures, and payload discipline across dependent services [7]. Third, it should preserve a synchronous API path for use cases that depend on direct request-response behavior, such as label generation, carrier-rate lookup, or appointment confirmation workflows [1]. Fourth, the architecture benefits from modular service boundaries that allow execution support, analytics, optimization, and user-facing coordination tools to evolve without destabilizing the transactional core [5; 10]. S/4HANA remains the system of record for order, delivery, billing, and freight-related business objects. The event-driven API layer externalizes execution states, carrier interactions, tracking updates, and exception workflows into a governed integration layer.

The conceptual model is shown in Figure 1. It adapts the interplay between synchronous and asynchronous interactions described in service and event-mesh research [1] to the needs of SAP-centric multi-carrier shipping, while incorporating the loose coupling and actor integration logic discussed in collaborative logistics and production-logistics architecture studies [9; 10].

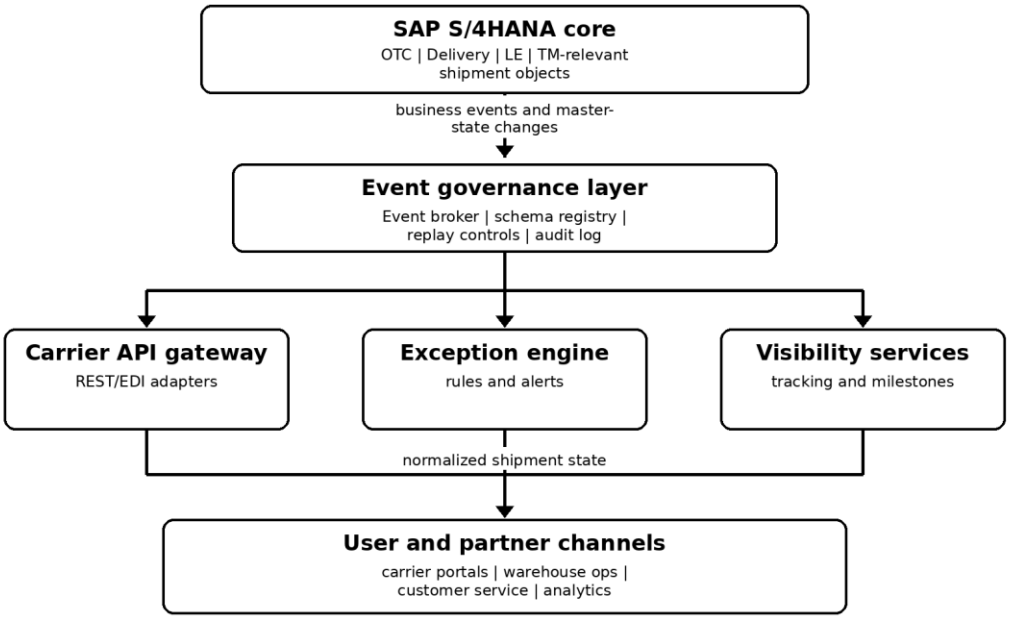

Figure 1: Conceptual event-driven API architecture for SAP-centered multi-carrier shipping (adapted from Calvio and his colleagues [1])

Shipping integration combines rapid state propagation with tightly controlled handling of label generation, carrier-rate lookup, and appointment confirmation [1]. Khriji and his colleagues show why an event backbone improves decoupling and responsiveness under heavy data flow [6]. Microservice evidence warns that such benefits disappear when traceability of event origin and processing path, ordering logic, and schema discipline are neglected [7]. Operational logistics research brings the discussion back to practice by showing that bidirectional real-time coordination is increasingly tied to modular service ecosystems and web-native orchestration layers [5]. For SAP logistics, carrier onboarding should rely on canonical events and standardized APIs, while document status, transport milestones, and exception triggers stay under explicit audit-ready control. In enterprise shipping environments, that integration layer also has to govern EDI and IDoc flows, external carrier APIs, and outbound or inbound shipment-status messages that cross the SAP boundary yet still affect delivery, billing, freight execution, and credit-related OTC control. Implementation logic during modernization depends on rollout order, governance design, and control over live carrier traffic. The surveyed authors reject abrupt architectural substitution. Transformation research emphasizes the need for an explicit strategic frame that links digital architecture to organizational direction, process redesign, and capability development [8]. Production-logistics architecture studies point in the same direction by treating digital services as value-oriented layers that improve monitoring and decision support around core operations [10]. Freight-platform research shows that service

modularity matters because platform functions mature unevenly across visibility, optimization, and analytics [4]. Shipment-tracking research offers a grounded reminder that document handling, waiting times, and workload reduction become visible only when external interaction patterns are redesigned alongside the supporting information flows [3]. In an SAP program, migration should therefore begin with event identification for shipment-state changes and cross-boundary exceptions, continue with carrier API normalization and observability, then extend to analytics, prediction, and digital twin functions. This order limits disruption in live execution and keeps S/4HANA conversion readiness aligned with real execution priorities.

Migration sequence matters because loose coupling, continuous adaptation, transformation strategy, and event governance must develop along the same rollout path. Xu and his colleagues extend that argument to multi-party coordination and explain why loosely coupled architectures suit actors that hesitate to share data through a tightly integrated monolith [9]. Dynamic Freight Management research pushes the same point toward continuous data-driven adaptation in volatile transport environments [2]. Digital transformation research warns that technology initiatives lose momentum when disconnected from a coherent transformation strategy [8]. Event-management research, by contrast, exposes the operational debt created when teams introduce asynchronous communication faster than they develop governance for schemas, observability, and recovery [7]. Event expansion requires schema control, replay policy, and observability, while API standardization and digitized carrier touchpoints require end-to-end event visibility and named ownership across functional design, integration engineering, and AMS.

4. Discussion

Real-time shipping modernization assigns transactional control to S/4HANA and routes execution visibility, carrier messaging, and exception circulation through a governed event-driven layer. The event-driven API layer is responsible for distributing operational state, partner-facing service contracts, carrier-neutral API contracts and state mapping across carriers, exception circulation, and rapid feedback into operational workflows.

Table 1 contains the implementation sequence, architectural focus, SAP-facing deliverables, and exit criteria for each phase.

Table 1: Phased implementation logic for SAP-centered multi-carrier shipping modernization

Phase	Primary design task	SAP-centered deliverable	Integration outcome	Exit criterion
1. Process baseline established	Map OTC, delivery, freight, carrier, and exception flows.	Event inventory linked to business objects and status changes.	A clear line drawn between core transactions and external interactions.	Canonical list of shipment events approved.
2. Canonical integration model.	Define event taxonomy, API contracts, payload ownership, and state transitions.	Functional specifications for business events and carrier-neutral APIs.	Reduced interface ambiguity across carriers.	Contract set validated against representative carrier scenarios.
3. Connectivity layer	Build API gateway, adapter pattern, security controls, and routing logic.	Reusable integration components for carrier onboarding.	Faster onboarding and fewer point-to-point dependencies.	First carrier group connected through standardized contracts.
4. Visibility and control	Add observability, replay, alert rules, and reconciliation logic.	Audit-ready monitoring design with operational dashboards.	Traceable shipment state and earlier exception detection.	Event traceability verified; replay and recovery procedures tested.
5. Stabilization and scale	Embed AMS governance, change control, and performance reviews.	AMS stabilization package with L2/L3 runbooks, root-cause workflow, schema-change policy, break-fix documentation, and release cadence.	Sustainable operations after go-live.	Manual interventions and incident recurrence decline.

Shipping modernization breaks down when teams build connectors before agreeing on event semantics and expose APIs before defining reconciliation across SAP, carriers, and monitoring services. The phased model avoids that trap. The approach prioritizes semantic consistency over technical complexity, and operational control over expansion. Evaluation should combine architectural soundness with performance during execution.

To fully assess a building's quality, it's essential to consider both its structural soundness and how well it performs in practice. Table 2 lists the metric families used to separate operational value from raw integration traffic.

Table 2: Monitoring metrics for event-driven multi-carrier shipping systems

Metric family	Indicator	Operational meaning	Review cadence
Event timeliness	Median event propagation time	Speed of shipment-state movement across systems	Daily
Delivery of signals	Event acknowledgment success rate	Reliability of outbound and inbound message handling	Daily
State integrity	Reconciliation mismatch rate	Divergence between the SAP state and the carrier or visibility state	Daily
Exception handling	Mean time to detect and route exceptions	Responsiveness of rule-based operational control	Weekly
Manual dependency	Manual touch rate per shipment	Residual reliance on human correction or spreadsheet work	Weekly
Change resilience	Failed deployments caused by contract or schema changes	Stability of integration governance during evolution	Monthly
Carrier scalability	Average onboarding lead time for a new carrier	Reusability of the integration model	Monthly
Audit quality	Percentage of shipments with a complete event trail	Readiness for regulated review and post-incident investigation	Monthly

The metric set links latency, reconciliation quality, onboarding speed, workload, and schema stability to observable operating results. Event latency requires joint reading with reconciliation quality; low propagation time loses analytical value once mismatch rates rise. The combined metric set, therefore, prevents one-dimensional success claims. It ties technical behavior to workload, visibility, and governance.

Ownership needs a permanent governance nucleus between SAP functional design, integration engineering, and AMS operations. Without that nucleus, canonical events drift, carrier-specific exceptions accumulate in undocumented logic, and support teams inherit a landscape whose behavior no longer matches its design

documents. That governance layer needs named ownership for schema changes, root-cause analysis for failed event propagation, L2/L3 stabilization routines, and reference documentation that survives carrier onboarding waves and post-go-live support. In long S/4HANA programs, that drift is what turns integrations into operational liabilities. The event model merits treatment as a controlled business asset because shipment-status semantics govern execution decisions across teams and partners.

5. Conclusion

Legacy multi-carrier shipping underperforms when shipment-state data fragments across ERP, carrier interfaces, and visibility services. A workable modernization path keeps S/4HANA as the transactional system of record and places carrier contracts, event publication, exception routing, and observability in a governed event-driven API layer. Migration begins with event inventory and canonical contract design, proceeds through carrier-neutral APIs and observability, and ends with AMS governance that prevents new integrations from reproducing legacy fragmentation.

Acknowledgements

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1]. Calvio, Alessandro, et al. “An Event- and Service-Mesh Architecture Supporting Service Integration in Society 5.0-Enabled Smart Cities.” *Proceedings of the 2023 ACM Conference on Information Technology for Social Good (GoodIT ’23)*, Association for Computing Machinery, 2023, pp. 462–470. https://doi.org/10.1145/3582515.3609568

[2]. Galkin, Andrii, et al. “Navigating the Future of Urban Logistics: Conceptual Framework for Dynamic Freight Management.” *Transportation Research Part D: Transport and Environment*, vol. 147, 2025, p. 104956. https://doi.org/10.1016/j.trd.2025.104956

[3]. Habjan, A. “Real-Time Shipment Tracking through IT-Enabled Systems and Electronic Logistics Marketplaces.” *Business Systems Research Journal*, vol. 16, no. 2, 2025, pp. 43–68. https://doi.org/10.2478/bsrj-2025-0018

[4]. Heinbach, C., et al. “Data-Driven Forwarding: A Typology of Digital Platforms for Road Freight Transport Management.” *Electronic Markets*, vol. 32, 2022, pp. 807–828. https://doi.org/10.1007/s12525-022-00540-4

[5]. Ieva, S., et al. “Enhancing Last-Mile Logistics: AI-Driven Fleet Optimization, Mixed Reality, and Large Language Model Assistants for Warehouse Operations.” *Sensors*, vol. 25, no. 9, 2025, p. 2696. https://doi.org/10.3390/s25092696

[6]. Khriji, S., et al. “Design and Implementation of a Cloud-Based Event-Driven Architecture for Real-Time Data Processing in Wireless Sensor Networks.” *The Journal of Supercomputing*, vol. 78, 2022, pp. 3374–3401. https://doi.org/10.1007/s11227-021-03955-6

[7]. Laigner, Rodrigo, et al. “An Empirical Study on Challenges of Event Management in Microservice Architectures.” *ACM Transactions on Software Engineering and Methodology*, 2025, https://doi.org/10.1145/3776581

[8]. Wurm, B., et al. “A Revised Framework for Digital Transformation Strategies: Contemporary Insights and Future Research Pathways.” *Electronic Markets*, vol. 35, 2025, article 99. https://doi.org/10.1007/s12525-025-00838-z

[9]. Xu, Liming, et al. “Multi-Agent Digital Twinning for Collaborative Logistics: Framework and Implementation.” *Journal of Industrial Information Integration*, vol. 45, 2025, p. 100799. https://doi.org/10.1016/j.jii.2025.100799

[10]. Zafarzadeh, M., et al. “A Framework and System Architecture for Value-Oriented Digital Services in Data-Driven Production Logistics.” *International Journal of Production Research*, vol. 63, no. 18, 2025, pp. 6648–6668. https://doi.org/10.1080/00207543.2025.2480204