Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

# An Automated Integrated Framework for Purchase Order Exception Handling and Manufacturing Synchronization

**Shobhanjaly P.Nair**

Assistant Professors,Department of Computer Science and Engineering,Sri Venkateswara College of Engineering, Chennai, Tamil Nadu, India.

**Abstract**

Today's factory supply networks depend on ERP platforms to link buying information with workshop activity. Yet POs often fail checks or layout rules when divided or turned into barcodes. Because these steps do not sync well, delays hide what is happening, leaving outside suppliers unaware of current tasks - spreading issues across the chain. A robust, self-driven cyber-physical model is introduced here, built on separate layers of small services that spot broken purchase records, contain them, then fix without help.Starting with Oracle Integration Cloud, the system handles real-time coordination between services while using Oracle APEX to track diagnostics from one main hub. Instead of relying on batch updates, live signals from the vendor interface feed directly into a master record stamped with barcodes. That running log then kicks off event transfers through Apache Kafka, which stores each change in permanent sequence. Because every update flows through this chain, nothing gets lost along the way. Recovery takes less time now since errors are caught early and fixed automatically. Manufacturing sites spread across locations stay aligned without extra help from central IT teams. Compliance stays intact because every step leaves a traceable mark.

**Keywords—**Supply Chain Resilience, Digital Twin, Oracle Integration Cloud, Apache Kafka, ERP Exception Handling, Event Streaming, Cyber-Physical Systems.

## 1. INTRODUCTION

In enterprise-scale manufacturing ecosystems,the factory requires purchase orders for buying goods because the process needs to maintain proper product division until the items reach their destination. The suppliers use these split components because they create individual barcodes which allow them to track each shipment from delivery to billing process. Although Oracle Fusion SCM tools have developed advanced features over time, the system still encounters delays when users attempt to connect central business records with external vendor websites. The system encounters problems when two systems attempt to exchange information because their synchronization schedules do not operate correctly.

A purchase order might stall if details mismatch, key info is absent, or connections drop too soon. The system remains suspended in this state because ERP systems failed to complete its processing while they need to show its active status. The absence of detailed logs which track header failures and error occurrences leads to teams losing visibility on their work progress. Suppliers sit idle, unable to start tasks

because updates fail to show in their dashboards. The production schedule faces delays because the production process reaches a halt point, which extends into creating new production schedules**[1].**

IT personnel previously needed to resolve sync problems through their physical presence at the work site. DB administrators conducted table inspections manually because the system lacked automatic correction capabilities. The team located defective transaction numbers, which they used to execute their work procedures through a gradual process. The approach which handles issues after their occurrence creates situations where people can make errors. The needed work to resolve delays required both visual examination and physical contact with the equipment. The system maintained synchronized data gaps longer than necessary. The current data regulations require organizations to maintain complete tracking mechanisms. The absence of ongoing documentation which records the timing of all operational changes creates difficulties for businesses. The absence of a documented timeline which tracks every operational procedure leads to regulatory problems.

The new solution establishes a sustainable framework through a dual cyber-physical system that combines multiple distinct systems. The system operates through its current mode of operation which prevents errors from occurring.

## 2. Literature Review

### a. Digital Supply Chain Ecosystems and Digital Twins

The conceptual foundation comes from the "Digital Supply Chain Ecosystem" (DSCE). According to Ivanov [1], this ecosystem is a network connected by digital tools, AI, and cloud computing, allowing live models to respond in real time. This project builds on the work of Grieves and Vickers [5], who described the Digital Twin as a tool to reduce unexpected behavior through digital mirroring. We use Level 3 "Intelligent Digital Twins". Unlike passive Level 1 monitoring, these copies adjust automatically using a middle layer to restart stuck purchase orders. Tao et al. [6] highlight that the industrial use of digital twins must provide a "digital thread" to ensure data integrity throughout the entire production lifecycle.

### b. Risk Interaction and Failure Propagation

Because systems often break in one area and then fail everywhere, studies now follow how small data issues spread inside factory grids. Liu et al. [2] introduced "Higher-Order Interaction Networks" to track how data failures move through complex systems. When ERP setups connect too closely, a single mistake in checks can stop all supplier deliveries. This project applies this theory by using a JSON-based Audit Log and Kafka to create a "Traceability Node" that prevents the quiet failures noted in the literature.

### c. Intelligent Orchestration and Dynamic Scheduling

Modern manufacturing needs real-time synchronization instead of the old batch-based systems. Li et al. [3] show that using deep reinforcement learning to dynamically schedule machines is much more effective than using static planning for orders with multiple items. Instead of waiting hours for old-style bulk updates, live adjustments that are based on incoming signals work much better for mixed orders. This proves that Oracle Integration Cloud (OIC) can be used to respond to events as they happen, without having to wait for slow clockwork imports.

**d. Strategic Resilience and Event Streaming**

Following the ideas put forth by Berger et al. [4], modern supply chains need data logging that can't be changed. Monostori [7] says that Cyber-Physical Production Systems (CPPS) need to be able to deal with uncertainty at the local level in order to keep the world stable. Overwrites can happen to traditional databases. This framework fills that gap with a Barcode Trail Log and distributed event streaming (Kafka). Garg [8] shows that using Kafka to stream data in real time makes sure that every state change is saved as a permanent, sequential event.

## 3. Analysis and Problem Gap

Looking at recent studies [1]-[8], it turns out the idea behind supply chain digital twins is solid on paper - yet real-world use still lags. Most academic work frames resilience either as broad strategy or tangled math puzzles. What's missing? Clear focus on how these models handle disruption in daily operations

1) **Practical Integration:** Putting ideas into action often hits a wall. Few setups offer clear steps for systems such as Oracle APEX. These tools struggle to fix mismatches in ERP checks. Guidance on actual setup remains sparse across most models. Connecting them smoothly? Rarely mapped out

2) **User-Centric Recovery**: Most tools today leave regular employees stuck when transactions fail. Getting things back on track often means waiting for tech teams to step in. Business workers who know the process best still need help from database experts. This delay slows everything down, even if the fix seems obvious. Fixing errors independently remains out of reach for those outside IT departments.

3) **Traceability in Transitions:** When moving data from a main system to a supplier platform, gaps appear. Visibility slips away mid-process, especially in delayed handoffs. Systems fail to sync right away. Information gets stuck between environments. This delay breaks the chain of tracking. Updates vanish without clear records. The gap hides what happens in transit.

This study addresses these gaps by proposing a decoupled, 4-tier architecture that provides both technical resilience and user-level observability.

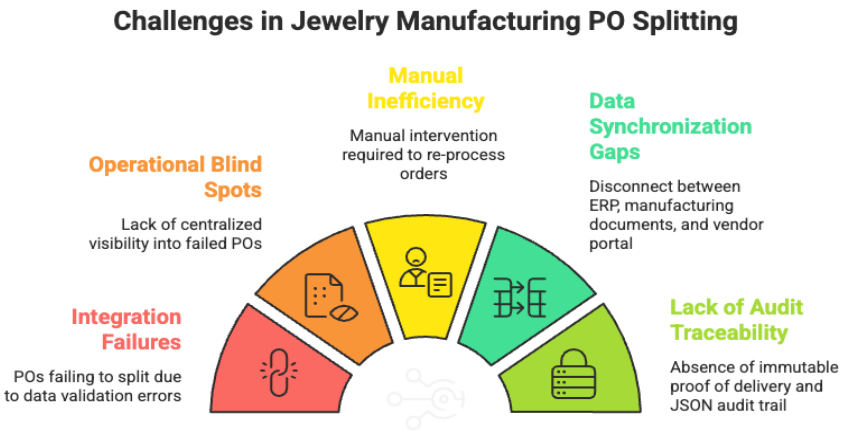


**Fig. 1. Issues & Challenges**

## 4. PROPOSED METHODOLOGY

The proposed system functions as an active, data-driven Digital Twin by employing a "Select-Retry-Sync" integration framework. As shown in the End-to-End System Data Flow (Fig. 1), the architecture is strictly decoupled into four highly specialized logical layers, which are further detailed in the Decoupled 4-Tier System Architecture (Fig. 2).To achieve high availability, fault tolerance, and separation of concerns, the architecture is strictly decoupled into four highly specialized logical layers:

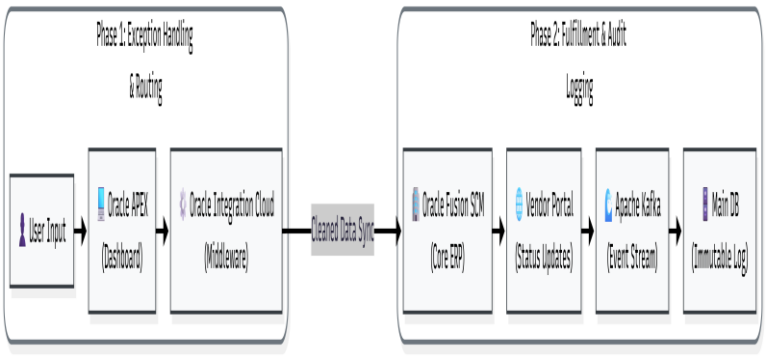


**Fig. 2. End-to-End System Data Flow from User Interaction to Immutable Audit Log.**

### e. Presentation and Diagnostic Layer (Oracle APEX)

The Presentation Layer acts as the primary observation and control node for supply chain managers. **The PO Split Error Dashboard (Fig. 3)** identifies Purchase Orders that failed the splitting process and surfaces them for user action. It is developed using Oracle APEX, providing a low-latency, web-based interface that securely connects to the underlying Oracle database.

**1. Error Resolution Dashboard**: This module serves as the primary diagnostic interface for identifying validation failures. Instead of making users write database questions themselves, it pulls purchase orders stuck in error status straight from temporary storage areas. Each entry appears with its ID number, moment it broke, and exactly why - no code needed to see it. Rows line up neatly so spotting trouble takes seconds. Behind the scenes, queries run automatically. This streamlined interface centralizes error visibility, allowing administrators to diagnose structural payload failures without requiring direct database queries

**2. Action Controllers:** Instead of waiting on IT staff, the APEX screen runs a built-in script that handles multiple selections at once - so people can restart failed processes all together. Once someone clicks retry, the system gathers just the needed purchase order details, wraps them into an encrypted data package, then sends it off through a standard web connection to the backend processor.

### f. Middleware Orchestration Layer (OIC)

Inside the system, Oracle Integration Cloud acts like a hub where data moves and adjusts. When errors occur, handling them happens here instead of inside the main accounting software. This separation keeps Oracle Fusion running smoothly without extra tasks piling up. Retries happen outside, so the heart of operations stays clean and focused.

**1.Protocol Translation:** Starting from the middle, OIC sets up two-way communication between REST and SOAP systems. One step at a time, it pulls in JSON messages sent by APEX without waiting for responses. Instead of skipping details, it reshapes each piece of information through graphical XSLT tools. Not simply converting, it reformats everything into exact SOAP XML that matches Oracle Fusion's needs.

**2.Idempotency and Logic Execution:** Even when connections drop, the system keeps working without errors. Splitting orders into smaller parts happens through smart rules inside OIC. When retries happen, nothing repeats by mistake. This setup stops extra purchase orders from appearing in ERP systems. Mistakes like double entries vanish because each step remembers what it already did.

**g. ERP Fulfillment and Synchronization Layer (Oracle Fusion)**

After OIC handles the split and cleans up the data, it moves forward through protected SOAP web services into Oracle Fusion Cloud SCM. The transfer kicks off only when both steps are fully complete. Security stays active throughout the entire transmission process. Each piece of information follows a strict path without deviation. Nothing gets sent until validation passes. Only verified chunks reach the destination system.

**1.System of Record:** Oracle Fusion acts as the ultimate system of record, generating specific Work Orders and assigning unique alphanumeric Barcodes for each split line item.

**2.Automated Synchronization:** To synchronize this newly generated data with external systems, PL/SQL background schedulers (DBMS_SCHEDULER) are deployed. These extract the barcode data and pushing it to the external Vendor Portal in near real-time, eliminating vendor idle time.

**h. Real-Time Audit and Traceability Layer (Apache Kafka)**

To mitigate compliance risks and ensure absolute transparency, the architecture features a dedicated, highly scalable Audit Zone.

**1. Barcode Trail Log:** When the Vendor Portal receives and processes the workload, the status updates are instantly captured in a centralized Barcode Trail Log interface. This log provides a highly visible tracker of the item's.

**2. Event Streaming:** The Barcode Trail Log acts as the definitive source of truth and triggers an Apache Kafka Producer. The Producer serializes the lifecycle event into a standardized JSON payload and publishes it to a dedicated Kafka topic

**3. Immutable Persistence:** A downstream Kafka Consumer Service reads this stream and writes the logs into the CL Indus DB. Because Kafka operates as an append-only distributed log, this establishes an immutable ledger of all system transitions, securing the enterprise against data manipulation.

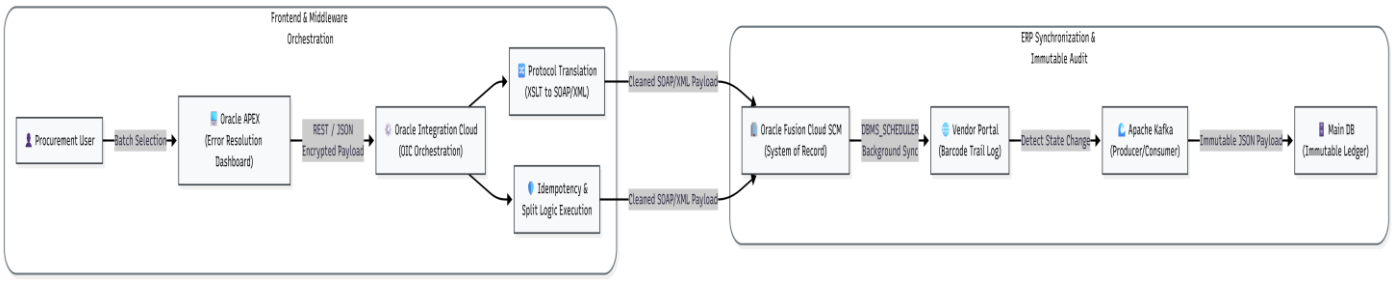


**Fig. 3. Decoupled 4-Tier System Architecture encompassing Presentation, Middleware, ERP, and Audit Layers.**

IMPLEMENTATION DETAILS

The proposed framework was realized using a targeted suite of enterprise tools: Oracle Database for data staging, Oracle APEX for the administrative frontend, Oracle Integration Cloud for API orchestration, Oracle Fusion SCM, and Apache Kafka for event logging. As shown in the **Barcode Trail Log (Fig. 5)**, every status modification from the Vendor Portal is captured and streamed asynchronously

### i. Mock Data Ingestion and Testing

Before going live, We fed fake purchase orders - showing complex production needs - into a temporary area of the Oracle database. Instead of real data, these entries carried flaws on purpose, shaped by regular PL/SQL code meant to trigger format issues and ERP rejections. Through this method, we watched how well the system caught mistakes and responded when users tried again. Errors appeared just as they would in reality, making it possible to study recovery steps closely.

### j. Middleware Routing and API Bridging

Messages move across separate layers only through standard REST and SOAP web services. Built into the OIC

level is a setup where triggers spark actions. Once the REST adapter gets a signal from the APEX dashboard, it fixes format issues inside the data. Then, by way of a SOAP adapter, the cleaned information reaches Oracle Fusion safely, skipping early checks that might block it.

### k. Real-Time Tracking Integration

The traceability mandate was fulfilled by connecting the Vendor Portal to the Kafka cluster. The custom Barcode Trail Log was programmed to detect any state changes on the portal. Upon detecting a change, it automatically formats a message and pushes it to a designated Kafka topic. This distributed streaming approach ensures that audit logs are reliably captured and stored, even during peak operational hours.

## RESULTS AND DISCUSSION

### l. Accelerated Incident Recovery

Before the change, fixing a PO split mistake meant long waits through slow IT requests and digging into databases by hand. Because of that, each broken batch sat stuck for anywhere from one to two days, leaving suppliers doing nothing. Shifting the fix process over to the OIC middleware layer cut delays sharply. With the Oracle APEX dashboard now pulling all alerts into one place, retrying many at once

became possible with just one tap.Due to this smoother setup, recovery time dropped fast - from hours down to few minutes.

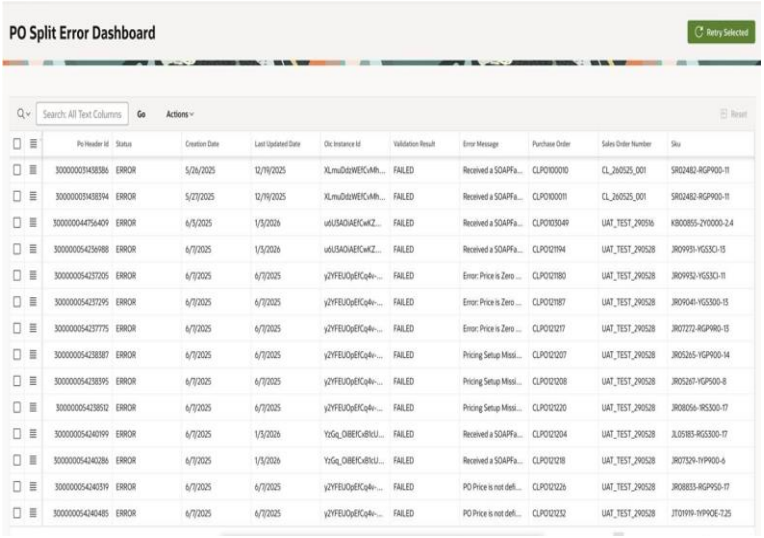


**Fig. 3. PO Split Error Dashboard With Retry-Sync Functionality**

### m. Mitigation of the Ripple Effect

When problems come up in purchase orders, they get spotted fast, so mistakes never travel further along. Work gets handed out smoothly to suppliers - barcodes arrive on time, keeping production moving naturally. Matching systems closely means the company records and vendor data stay perfectly aligned, stopping forecast mismatches before they start.

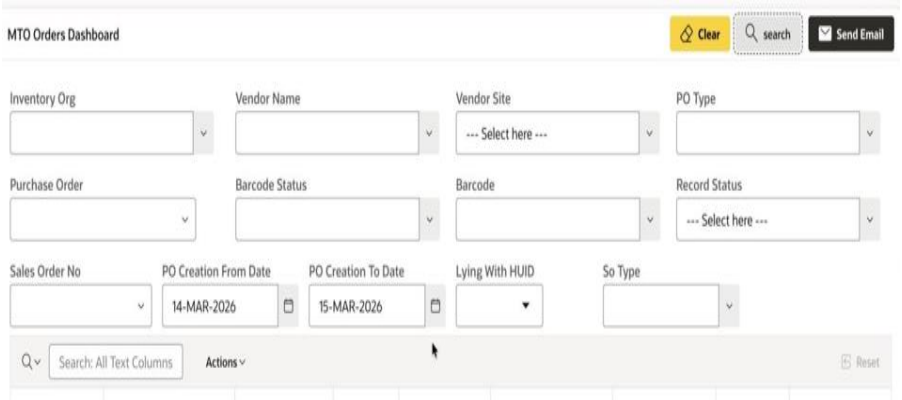


**Fig. 4. Vendor Portal for Vendor View of Purchase Order**

### n. Immutable Audit Compliance

The incorporation of Apache Kafka alongside the Barcode Trail Log- every shift shows up clear. Not just logged, but locked down for good in a separate system, each move from breakdown to approval stays visible. A silent copy sits apart, making audits solid by default. No more gaps when someone steps in off the books; every hand that touches it gets noted somewhere safe. Trust builds because nothing slips through unseen anymore.

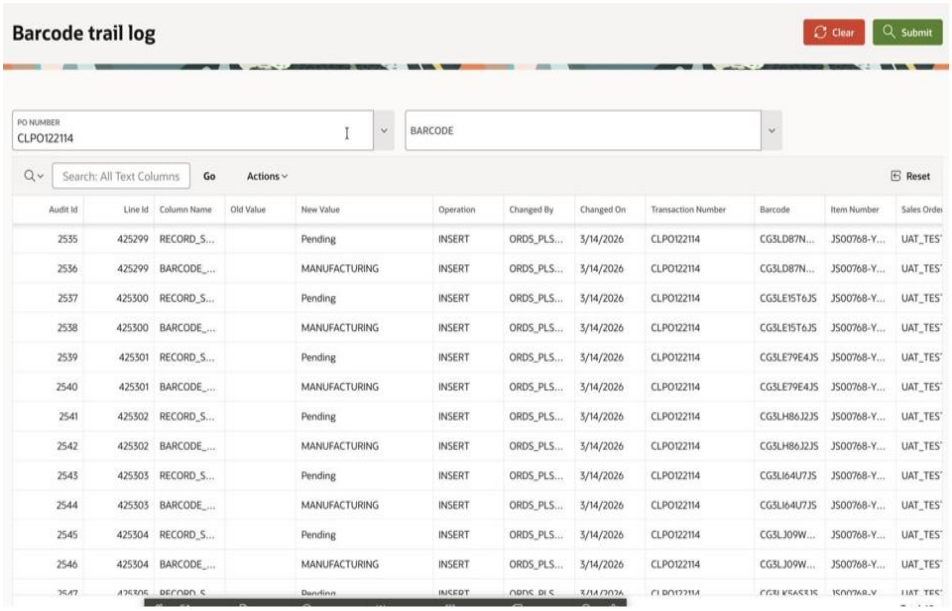


**Fig. 5. Barcode Trail Log Status Modification from the Vendor Portal**

## CONCLUSION

The new system made sure purchase orders move correctly and quickly from company software to factory networks, even if problems happen. Instead of failing, it keeps working. Using Oracle Integration Cloud, information flows in a smarter way. Old methods often got stuck fixing mistakes - this one does not. The setup cuts delays by managing steps more smoothly.

One big plus is that Oracle APEX puts diagnostic tools right in the hands of business teams, so they fix integration hiccups themselves - no DBA needed. What helps even more is how Apache Kafka streams events while the Barcode Trail Log locks in compliance, making every move visible. Because of this setup, one mistake won't flow in the system; everything stays aligned across the supply chain.

## DECLARATIONS

Funding: Not applicable. The authors received no specific funding for this work. Conflict of Interest: The authors declare no competing interests.

**Ethics:** The datasets used in this study are publicly available benchmark collections. All images used for evaluation are either synthetically constructed or drawn from publicly released research datasets and do not correspond to identifiable real individuals beyond the declared scope of each dataset.

## REFERENCES

1. D. Ivanov, "Conceptual and formal models for design, adaptation, and control of digital twins in supply chain ecosystems," Omega, vol. 137, p. 103356, 2025.
2. H. Liu, D. Shen, M. Dabić, and J. Lu, "A Novel Methodology for Risk Assessment Considering Risk Higher Order Interactions and Propagation Effects," IEEE Transactions on Engineering Management, vol. 72, pp. 907–924, 2025.

3. F. Li, S. Lang, and Y. Tian, "A Transformer-Based Deep Reinforcement Learning Approach for Dynamic Parallel Machine Scheduling," Journal of Intelligent Manufacturing, vol. 36, no. 2, pp. 112–128, 2025.
4. R. Berger, R. Wagner, P. M. Dion, and O. Matthias, "Disrupting Disruptions: Enhancing Supply Chain Resilience," Annals of Operations Research, vol. 347, pp. 1163–1192, 2025.
5. M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," Transdisciplinary Perspectives on Complex Systems, pp. 85-113, 2017.
6. F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," IEEE Transactions on Industrial Informatics, vol. 15, no. 4, pp. 2405-2415, 2019.
7. L. Monostori, "Cyber-physical production systems: Roots, expectations and R&D challenges," Procedia CIRP, vol. 17, pp. 9-13, 2014.
8. N. Garg, "Apache Kafka 101: A Beginner's Guide to Real-Time Data Streaming," Journal of Software Engineering and Applications, 2023.