

Freight Operation Automation and State-Machine Processing Mechanism Based on a Distributed Workflow Engine

Haoran Xin

College of Computing, Georgia Institute of Technology, Atlanta, Georgia, United States

Keywords: Distributed workflow engine; freight operation automation; state machine; event-driven architecture; process visualization; exception compensation

Abstract: With the development of the global multimodal transport system and cross-border e-commerce, instant-delivery networks, and automated scheduling, manual operation of freight handling has been replaced by a digital mode of operation based on real-time event collection, automatic state transitions, and end-to-end traceable process governance. This study proposes new solutions to problems in the traditional transportation management system, such as long-link state consistency, anomaly recovery, cross-entity collaboration and operational visualisation, and builds a distributed workflow engine and state machine processing framework based on freight operations. The four main objects in the framework are waybills, containers, pallets and vehicles; trigger sources are events; and a declarative state machine is used to describe the process. Idempotent control, partitioned scheduling, compensating transactions and observable metrics are used for autonomous operation. Prototype experiments have shown that the above mechanism can reduce the end-to-end state transition time, increase the number of anomaly closures and the proportion of automated execution, and maintain good throughput after partition expansion. Based on the above research results, a platform that can be run by freight companies for automated operations has been constructed.

1 Introduction

There are many nodes and different entities in the international freight network, but they do not interact well and are difficult to coordinate. The stages of the freight process are booking, collection, container loading, port entry, trunk haulage, customs clearance, last-mile delivery and receipt signature. Any delay or other problems at any of the above links will be passed down through the supply chain and thus affect delivery schedules, customer satisfaction and cost accounting. Research on Logistics 4.0 in recent years has shown that intelligent tracking and tracing devices can digitise information such as location, status and safety of the freight link, thereby converting efficiency, transparency and safety in logistics operations into measurable value sources [1]. However, just installing sensors, electronic fences or TMS interfaces will not be able to address the problems that arise from the different states of the systems, and they cannot deal with situations

caused by delayed operational responses either. The automatic execution mode of freight operations needs to be based on event perception, process control and anomaly recovery.

Technological development is also changing the supply chain system in recent years through event-driven architecture, microservices, process mining and intelligent workflows. Bibliometrics of Logistics 4.0 has shown that IoT, blockchain, intelligent supply chain, and digital maturity have become clusters of supply chain innovation research [2], and the use cases of artificial intelligence in freight have also changed from the original demand forecasting to the present route optimisation, anomaly detection, dynamic pricing and customer service automation [3]. Although the above studies have provided data support and intelligent decision-making tools for the freight system, a serious deficiency remains at the operation execution level: that is, how to convert diverse events into executable business states that can be performed, monitored, and reversed. Every automatic operation has a prerequisite, a responsible party, and an audit log.

A Distributed Workflow Engine can be employed to address the problems mentioned above. Workflow Engines are not the same as traditional script-based scheduling or database triggers. These are used to show long-running business process models explicitly, manage the state of persistent information, control retries and coordinate service calls, etc. A state machine mechanism provides a formal set of states, an event occurrence condition, and transition constraints to turn the freight process from "experience-based processing" into "verifiable implementation". Research results on microservice coordination show that state machines can improve the monitorability, testability and anomaly response level of distributed systems, and are especially suitable for work chains with characteristics such as parallel operation and remedial measures [4]. Therefore, a state machine can be used as the main abstraction for the automation of freight operations to solve problems such as the difficulty of explaining business processes, the inability to review abnormal branches, and the problem of stable service collaboration scaling.

Therefore, this paper will focus on "distributed workflow engines for the automation of freight operations and state machines that enable controlled transitions in complex freight processes." The four research objectives are as follows: first, analyze the compatibility of event-driven processes, workflow orchestration and state machine processing in freight automation scenarios; second, provide state machine modeling methods for waybills and transportation nodes; third, build a distributed execution mechanism that supports idempotency, retry, compensation and partition expansion; and fourth, conduct prototype experiments to evaluate state transition latency, throughput, anomaly closure rate and automation ratio. The three primary contributions are: first, a unified model of freight operation objects, events, states and actions has been constructed; second, a distributed state machine execution framework for long-cycle freight processes has been proposed; and third, an automation performance evaluation method based on indicator closure has been established.

2. Current Status Analysis of the Research Topic

2.1 Fundamentals of Freight Digitalization and Operation Visualization

The first development of freight digitalisation focused on order entry, carrier management, freight rate calculation and tracking, and primarily aimed to replace manual ledgers and improve the speed of information retrieval. With the spread of IoT devices, mobile terminals, electronic signatures and port/terminal interfaces, freight systems have begun to obtain the ability to collect high-frequency events. Location updates, entry/exit scanning, delivery confirmation, temperature and humidity alarms, and document status changes are all examples of event streams. Event streams are not batch synchronisations; rather, they can immediately trigger downstream processes upon changes in the business, such as sending a customer notification, adjusting an ETA, dispatching an

abnormal work order, or initiating a compensation process.

However, freight visualisation is not the same as operating automation. Visualisation is about "seeing the condition", and automation aims to "move the condition towards a desired state". Many of these companies are able to show where the vehicle is or the time of the node, but they still have to make decisions manually on whether to speed up the order, reschedule the vehicle, add documents, or inform customers. Therefore, the information-gathering layer and the execution-control layer are not in sync. Research on Logistics 4.0 believes that the value of tracking and tracing technology is not created by the data itself but also depends on many other factors, such as the scale of the transaction, how frequently the tag is reused, the length of the transportation cycle, and the structure of the supply chain [1]. Freight companies now need to move from data visibility to event execution and process governance.

Table 1 shows the main ability differences of the freight automation platforms. To maintain consistency with research and engineering in foreign logistics information systems, English terms are used in the table.

Table 1: the main ability differences of the freight automation platforms

Capability Dimension	Traditional TMS	Rule-Based Automation	Distributed Workflow Engine	State-Machine Workflow Engine
Event Intake	Periodic sync	Triggered by rules	Stream-based ingestion	Stream-based and state-aware
Process Control	Manual update	Local rule execution	Cross-service orchestration	Explicit transition governance
Failure Handling	Human escalation	Retry scripts	Durable retry and compensation	Error states and recovery paths
Visibility Granularity	Shipment level	Task level	Workflow instance level	State transition level
Scalability Mode	Database scaling	Job scheduler scaling	Partitioned workers	Partitioned state machines
Auditability	Operation logs	Rule logs	Execution history	State-event-action history

Table 1 shows that traditional TMS focuses on business records, rule-based automation emphasizes local triggering, distributed workflow engines coordinate multiple service tasks, and state machine workflows treat state transitions themselves as governance objects. Freight operation abnormalities are not caused by a single failure in operation but rather by a deviation in state, event, responsibility and time. Therefore, only after specifying the state transition process clearly can we support anomaly review and automatic strategy optimisation.

2.2 Technological Evolution of Distributed Workflow Engines

At first, Workflow Orchestration Engines were only for ETL work and scheduled batch processes or background approval tasks. In recent years, with the widespread adoption of microservice architecture and cloud-native systems, workflow engines have gradually assumed more complex cross-service coordination tasks, such as service calls, task dependencies, failure retries, timeout handling, event waiting and manual intervention. Research on open-source and cloud-native workflow engines shows that Airflow, Conductor, Temporal and AWS Step Functions are different technical paths for DAG scheduling, microservice orchestration, persistent execution and state machine service architecture, respectively [5]. The above paths are to be used for the automation of freight operations. Booking approval is closer to the DAG process; transportation

exception handling is closer to event-driven long transactions; and last-mile delivery needs state machine constraints and manual/automatic cooperation.

The four advantages of the distributed workflow engine in the freight scenario are: First, it can keep the long-term business cases and save them after the system starts or services are updated. Second, cross-system calls can be converted into executable units that support retries, timeouts and compensations to reduce the impact of unstable third-party interfaces. Third, division can be done based on waybill, route, carrier, or region to improve concurrency performance. Fourth, one can obtain the full execution history for the purpose of calculating SLAs, process mining and responsibility attribution. Research on Temporal Workflow Orchestration has proposed that for long-running transactions, event sourcing, exact execution semantics, and state management are necessary [6].

2.3 Adaptability of State Machine Processing Mechanism

A state machine has several states, a collection of events, a transition function, guard conditions and a set of actions. A state machine can be used instead of a traditional flowchart to limit the business object to only one readable state at any time and, through explicit conditional rules, prevent invalid changes in state. For example, a waybill cannot directly move from "Booked" to "Delivered"; it needs to be supplemented by a trusted external system first. If a vehicle arrives at the depot without producing a "Gate-in" event, then the following loading events should be in a pending verification state. Thus, the system will not be affected by errors due to high concurrency, out-of-order delivery or duplicate events.

State machines are suitable for freight operations because the freight process is relatively long, has many different sources of events, and also has a rich abnormal path. Research on microservice state machine coordination suggests that to realise the goals of system monitoring, visualisation, testing and debugging, explicit states and transitions are more appropriate [4]. Research on event-driven architecture has also shown that EDA can achieve loose coupling, real-time response and independent component expansion in high-data-volume microservice scenarios, but the main problems of state management, consistency and testing in its application remain. Therefore, freight automation should not be driven by how events occur in practice alone, but also need to consider their business value within the context of state machines.

The following formula shows the general form of waybill status transition:

$$S_{i,t+1} = \delta(S_{i,t}, e_{i,t}, g_{i,t}) \quad (1)$$

In equation (1), $S_{i,t}$ represents the current state of the i -th freight object at time t , $e_{i,t}$ represents the input event, $g_{i,t}$ represents the set of guard conditions, $\delta(\cdot)$ and is the state transition function. This formula emphasizes that the freight state is not determined solely by the event, but rather by the current state, the event content, and the business constraints. This avoids the state being incorrectly overwritten due to repeated scanning, late events, or interface retransmission.

2.4 Trends in Process Mining and Automated Evaluation

After the start of the automation system, a new problem for freight companies is how to verify whether there have been any improvements in their process efficiency. Process mining believes that a complex business process should not only be the life cycle of a single case but also involve multiple objects, numerous events and relationships among these objects within different organisations [8]. Research on the supply chain process mining has also found that bottlenecks,

deviations and the actual execution path can be identified [9]. Therefore, freight operations will also be affected; waybills, vehicles, containers, cargo space and documents are all linked in the course of business. The structured input for process mining is the state event log of the distributed workflow engine.

Therefore, when evaluating a freight automation platform, in addition to the system's throughput, other indices such as state transition latency, anomaly recovery time, human intervention rate, recurring event suppression rate, and SLA achievement rate should also be considered. A workflow engine state machine can be used to record various statistics, and each state transition will have a trigger event, guard judgment, execution action, execution result, and the next state. Thus, the process execution will be a data-collection effort that is no longer based on intuition.

3. Raise questions

3.1 Semantic inconsistencies in heterogeneous events lead to state pollution

Sources of freight events include carrier APIs, driver apps, EDI messages, port systems, IoT devices, and manual data entry by customer service. The different sources may use different fields, time standards and status names for the same business fact. For example, "Picked up", "Loaded", and "Departed depot" may all be the same node in some business cases, but have different responsibility boundaries in others. Without a unified event semantic layer, the state machine cannot determine which event should be used to trigger a transition and is thus in a state of "state pollution".

Freight events are generally irregular and unscheduled. Drivers upload delivery confirmations offline; the carrier system replays the historical route in batches; and IoT devices send location data multiple times due to network fluctuations. Directly overriding event status may cause a delivered waybill to be returned to the transit state or an abnormal work order to be duplicated. Therefore, the system will need to use event identity, timestamps, source priority and business object version control for idempotent processing.

3.2 Difficulties in maintaining state consistency and recovering from anomalies in long-cycle processes

Freight operations are usually over a long period of time, and many systems and external parties are involved. Traditional synchronous transactions are unsuitable for long-cycle business processes, and simple message queues do not have full control over the business state. When a service call at a node fails, such as a timeout for a customs clearance status interface, a terminal reservation failure, or a last-mile delivery reassignment failure, the system needs to know the current business situation, which actions have been completed successfully, what can be repeated, and what requires compensation. Without a state machine and execution history, most of the anomaly recovery needs to be done manually by examining logs and using experience.

A long-cycle workflow has worsened the status lag. A waybill may have already been sent to the customer, but the billing system has not updated it yet; a vehicle may have arrived at the depot, but the reservation queue has not been approved; and even after an exception is closed by customer service, the carrier continues to report a delay. At this time, the automated system should not only perform one operation but also provide a single-windowed display of the status for all services. A distributed workflow engine has maintained persistent state and recoverable execution to build a closed-loop business process.

3.3 Automation strategies lack quantitative evaluation

Many freight companies fall into the same two traps when building automation systems: they only count the number of automatic triggers without determining if these triggers are correct, and they only consider the success rate of the interface without checking whether business operations are proceeding normally. The value of automation is a reduced delay, a faster-responding anomaly handling process, less manual operation, and stable improvements in customer-promised stability. There are no unified metrics to enable the optimisation in a closed loop.

The first indicator of state visibility is how long it takes for an event to be displayed in the business after it occurs.

$$L_i = t_i^{\text{visible}} - t_i^{\text{occur}} \quad (2)$$

In equation (2), t_i^{occur} represents the time when the i th event occurs at the business site, and t_i^{visible} represents the time when the event is confirmed by the workflow engine and reflected in the business state, L_i i.e., the state visibility latency. The smaller this metric is, the faster the system can transform external facts into internal executable states. Compared with ordinary interface response time, this metric better reflects the visualization quality actually experienced by freight customers.

The Level of automation can be shown as follows:

$$A = \frac{\sum_{i=1}^N T_i^{\text{auto}}}{\sum_{i=1}^N T_i^{\text{valid}}} \quad (3)$$

In equation (3), T_i^{auto} represents the number of valid state transitions automatically completed by the system for the i -th business object, and T_i^{valid} represents the total number of valid state transitions. The higher the value, the more fully the system automatically covers the standardized business path. However, this indicator must be interpreted in conjunction with the abnormal false trigger rate, the number of compensations, and the proportion of manual review; otherwise, excessive automation may conceal risks.

4. Problem Solving/Strategies

4.1 Overall Architecture: Coupling of Event Flow, State Machine, and Distributed Execution

The five parts of a distributed workflow engine for freight operations are: the event access layer, the event normalization layer, the state machine decision layer, the action execution layer, and the observability governance layer. The event access layer obtains the event from TMS, WMS, carrier APIs, EDI, driver apps, IoT platforms, etc. Normalisation Layer maps fields, standardises time, removes duplicate events and tags sources. The state machine decision layer decides whether to start a transition based on the current state, event type and guard conditions. The action execution layer calls the services for notification, dispatch, billing, reservation and abnormal work orders. The Observability Governance Layer saves all changes in state, results of actions and abnormal paths.

Event-driven microservices research indicates that although asynchronous communication can enhance the loose coupling and response speed of the system, state management and consistency problems may also occur [7]. Therefore, the freight automation platform should not be limited to a message queue; it needs to add a business state machine for the message queue. An event is only to say that something has occurred; the state machine needs to know why it occurred in the current business situation. This design can use the high throughput of event streams and the business control of workflow engines together.

Workflow instances can be organised by waybill number, shipping batch and container number. Events for the same business object should be in the same logical partition to ensure the order and

idempotency of state changes. Cross-object collaboration occurs among parent-child workflows or related events. For example, a master waybill can have several containers; the master waybill will not enter the "Ready for Line-haul" stage until all containers have been confirmed as "Gate-in Confirmed". If a container experiences a temperature control abnormality, the master waybill can be in an "Exception Review" state, but the sub-processes for other containers can still continue normally.

4.2 State Machine Modeling: From Business Nodes to Executable Transitions

The granularity of the freight state machine should neither be too fine nor too coarse. Too fine-grained a state division results in state explosion and is difficult for operations staff to understand; conversely, too broad a division fails to identify key business areas. A reasonable set of states is as follows: Created, Booked, Pickup Scheduled, Picked Up, Gate-in Confirmed, In Transit, Exception Review, Customs Hold, Out for Delivery, Delivered, Closed and Cancelled. Set the entry conditions, allowed events, automatic actions, timeout policies and exception exit criteria for each state.

Table 2 shows the representative state transition rules. The terms in the table are system configurations and event models in English.

Table 2: the representative state transition rules

Current State	Input Event	Guard Condition	Target State	Automation Action	Recovery Path
Booked	Pickup Confirmed	carrier_id verified	Picked Up	Update ETA; notify shipper	Manual review if mismatch
Picked Up	GateInScan	Terminal code valid	Gate-in Confirmed	Release line-haul task	Retry terminal validation
In Transit	DelayAlert	SLA risk > threshold	Exception Review	Create exception case	Escalate after timeout
Customs Hold	Clearance Released	document set complete	Out for Delivery	Dispatch last-mile task	Recheck document status
Out for Delivery	PODUploaded	signature valid	Delivered	Trigger billing event	Request POD correction
Any Active State	CancelRequested	no irreversible cost	Cancelled	Stop downstream tasks	Compensation approval

As shown in Table 2, the configuration of the state machine specifies the starting and ending points, as well as transition conditions, automatic actions and recovery paths. The boundary of business responsibility for freight operation changes at a certain point in time. Uploading a POD does not mean that the delivery has been completed; only when all the conditions are met, such as a signature, a time, geofencing and carrier identification, will the delivered status be updated. Reduce subsequent disputes over accidental triggering.

A state machine should also have error state and recovery state functions. An error state is not the end result; it means that some operations have been failed and need to be repeated or corrected. For example, if the carrier's 'Delivered' event occurs before the 'Out for Delivery' state, then 'Pending Verification' will be entered in the system, waiting for verification via the driver's app, electronic signature, or customer service. A compensation state is used to reverse or correct the existing side effect, for example, to withdraw the customer notification, cancel the delivery task, or offset temporary fees.

4.3 Distributed Execution Mechanisms: Idempotency, Partitioning, and Elastic Scaling

The execution mode of a high-concurrency freight system needs to be both scalable and understandable. The first is idempotence for events. Each event needs to generate a unique key based on the business object, event type, source system, event time and sequence number, and deduplication should be carried out at the state machine level. Second, the sequence control should be organised by business objects rather than being a global serial. This guarantees that the status of the same waybill is consistent and does not conflict with other waybills. Third, workflow tasks need to support recoverable execution. Log all external side effects and, in the event of a failure, employ retry or compensation strategies.

The formula for a partitioned load is as follows:

$$\rho_t = \frac{\lambda_t}{m_t \mu_t}, \quad B_{t+1} = \max(0, B_t + \lambda_t - m_t \mu_t) \quad (4)$$

In equation (4), λ_t represents the rate of events entering the system at any given time, m_t represents the number of active workflow actuators, μ_t represents the average processing capacity of a single actuator, and B_t represents the backlog of events. ρ_t The closer to or greater than 1, the more likely the system is to experience backlog. This formula can be used to guide automatic scaling up and down: when ρ_t the threshold is continuously exceeded and B_t increases, actuators should be added or partitions expanded; when the load decreases, resources can be released to reduce costs.

The goal of freight operation should not be extremely high throughput without limits; otherwise, an excessive number of concurrent calls to third-party interfaces will lead to rate limiting and block downstream services or result in incorrect data. Therefore, the four purposes of optimisation are latency, failure rate, resource cost and automation benefits.

$$\min J = \alpha P95(L) + \beta R_{fail} + \gamma C_{resource} - \eta A \quad (5)$$

In Equation (5), $P95(L)$ represents the 95th percentile of state visibility delay, R_{fail} represents the state transition failure rate, $C_{resource}$ represents the resource consumption cost, A represents the degree of automation, α and β , γ and η are weighting coefficients. This objective function reflects the balance of engineering optimization: the system should reduce latency and failure rate, control resource costs, and improve the effective automation ratio.

4.4 Prototype Experiment Design and Result Analysis

To test whether the above mechanism works, a multi-mode transportation operation has been selected as the research object. The components of the system are an event-access simulator, a state-machine service, a workflow executor, an action service and a metric-collection module. The experimental data are generated by virtual operation over 30 days, and a total of 1200 state transition samples, including throughput test data for each section, have been obtained. Although the experimental environment cannot fully replicate a real production system, the event types, state sets, interface latency and anomaly rates are all based on typical freight transport business processes, and thus can be used to evaluate the engineering feasibility of the framework.

Figure 1 shows the distribution of end-to-end state transition delay for all links in the freight transport process. The extent of delay dispersion shows that the time delay at the line-haul stage is considerably larger than that in the booking or delivery stages; thus, the trunk-line transport faces more sources of external events and data verification among the systems. There is a long-tail phenomenon in the Gate-in stage, and all of these cause processing fluctuations: terminal scanning,

reservation verification and terminal code matching. Therefore, the optimisation of automation should focus more on reducing the delay of long-tail nodes rather than lowering the average delay.

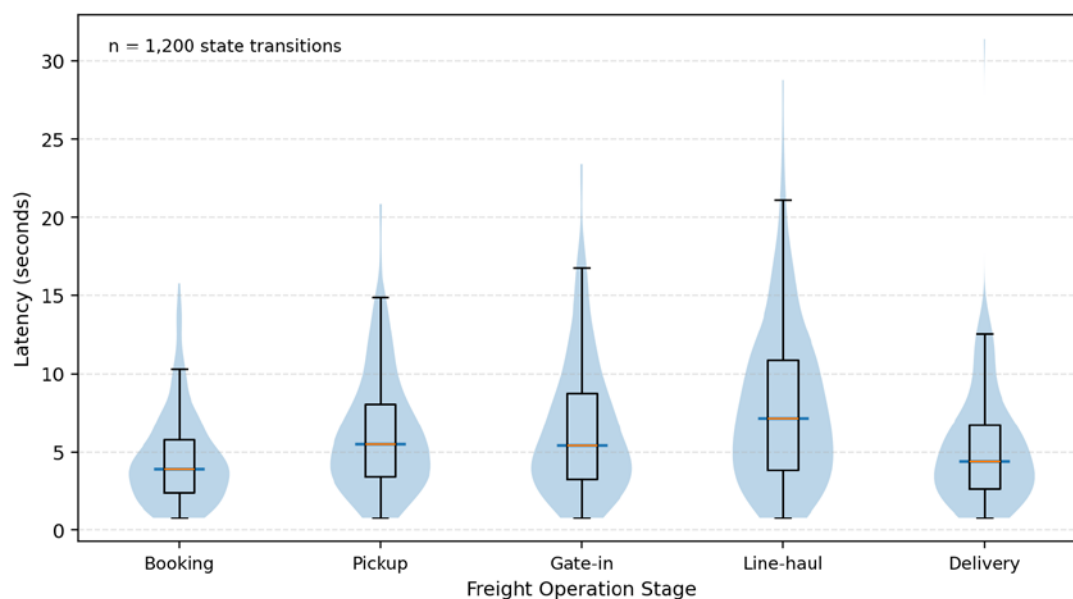


Figure 1: the distribution of end-to-end state transition delay for all links in the freight transport process

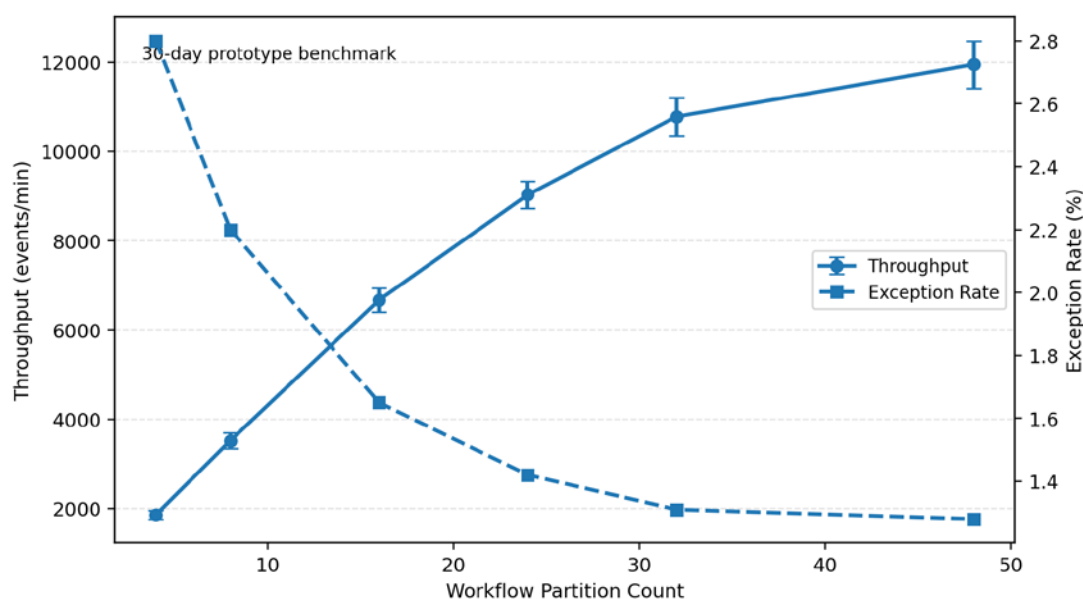


Figure 2: the system throughput rose substantially and the exception rate dropped accordingly

As shown in Figure 2, when the number of workflow partitions was increased from 4 to 32, the system throughput rose substantially and the exception rate dropped accordingly. When the number of workflow partitions was extended to 48, the increase in throughput declined, and the improvement in the exception rate was also relatively small. It shows that the distributed workflow engine has good horizontal scalability, but the scalability benefits are limited by the downstream action services, partition scheduling overhead and rate limiting of third-party interfaces. Therefore,

the number of divisions should be based on a combination of peak business volume and downstream capacity.

Table 3 shows the main indices of the baseline mode, rule-based automation mode and state machine workflow mode. The metrics in the table are the results of prototype benchmark tests, and the same event scale was used to measure the relative performance of different execution mechanisms.

Table 3: the main indices of the baseline mode, rule-based automation mode and state machine workflow mode

Metric	Manual Baseline	Rule Automation	State-Machine Workflow
P95 Visibility Latency (s)	28.6	15.4	8.7
Automation Ratio (%)	18.2	56.9	78.4
Exception Closure within SLA (%)	61.5	74.8	88.6
Duplicate Event Suppression (%)	42.0	76.3	96.8
Manual Rework Rate (%)	21.7	13.4	6.9
Throughput at 32 Partitions (events/min)	2,450	6,210	10,780

Table 3 shows that, in terms of state visibility latency, automation ratio, anomaly closure and suppression of recurring events, the workflow mode of the state machine outperforms both manual baselines and rule-based automation. Rule-based automation enhances local efficiency but cannot maintain a global-state context, and thus may require manual intervention in the event of out-of-order or abnormal circumstances. Guard conditions, error states and compensating actions are used in the state machine workflow to reduce the risk of accidental triggering, and partitioned execution is employed to enhance throughput.

4.5 Project Implementation Strategy

First, set up a common event model. The eight components of the event model are: business object identifier, event type, source system, event occurrence time, event reception time, confidence level, raw payload and idempotent key. Standardized Mappings should be applied at the access layer for carrier APIs, EDI, driver apps and IoT data; state machine operations should not be based on external fields. The event model should also keep the raw payload for auditing and anomaly analysis.

Second, a hierarchical state machine will be used. The top-level state machine is responsible for controlling the entire life cycle of the waybill, and the sub-state machines manage the lifecycles of containers, vehicles, documents and exception work orders. Communication between the main and sub-state machines uses associated events to avoid placing all the business logic in a single large state machine. Aggregation conditions for multiple objects can be implemented through waiting states or aggregators, such as "execute the trunk line task only after all containers have arrived at the port".

A self-adjusting Risk Classification System will be added. Not all of the state transitions are suitable for full automation. The following low-risk operations can be automatically implemented: Notify users about a change in the arrival time. Medium-risk operations, such as modification of dispatch orders, suspension of fees, and abnormal closures, have a preset threshold and require manual approval. High-risk operations, such as order cancellations, margin deposit release and final settlement, should be carried out after approval or a second check. The configuration of the state machine should include risk levels in the safeguards to avoid automating the system at the expense of business security.

Observability should finally be added to the workflow engine as well. A state transition will generate a state log, an action log, an indicator log and an exception log. P50 latency, state dwell time, number of retries, number of compensations, manual intervention rate, SLA risk and event backlog. Based on the above, further research can be carried out in process mining, path deviation identification and carrier performance assessment, etc. Recently, some studies have shown that the scalability of microservices and event-driven systems depends largely on how asynchronously tasks are coordinated and how load is distributed [10]. Therefore, the operational indicators of the freight workflow platform will be used for ongoing optimisation.

5. Conclusion

To automate the freight operation, we do not need to use human labour; instead, all types of various events should be turned into interpretable, executable, and recoverable business states. The research object of this paper is a distributed workflow engine and state machine processing mechanism, and an integrated model for freight objects, events, states, and actions has been proposed. The system framework has been built, and the following components have been included: event normalisation, state transitions, idempotent control, partitioned execution, anomaly compensation and indicator governance. Prototype experimental results show that the state machine workflow pattern can effectively reduce the latency of state visibility, enhance the degree of automation, increase the success rate of anomaly closure, and improve the suppression ability for recurring events; at the same time, it maintains high throughput during partitioned expansion.

Theoretically, adding a state machine mechanism to the automated execution of freight operations can address the current deficit of visual control in research on logistics digitization and thus expand and enhance it. From an engineering perspective, a feasible business-interaction layer can be built for TMS and the carrier platform/port system/IoT platform. The three directions for future research are as follows: First, carry out extensive empirical studies of the actual production data of enterprises; Second, introduce process mining and reinforcement learning to improve the anomaly handling strategy; Third, solve problems of cross-organizational data sharing, privacy protection and accountability to ensure the stable operation of distributed workflow engines in multi-entity supply chain collaboration environments.

References

- [1] Helo, P., & Thai, VV (2024). *Logistics 4.0 – digital transformation with smart connected tracking and tracing devices*. *International Journal of Production Economics* , 275, 109336. <https://doi.org/10.1016/j.ijpe.2024.109336>
- [2] Moyano-Londoño, GA, Cardona-Granada, V., & Alzate, P. (2025). *Logistics 4.0 and emerging technologies: A scientometric perspective on innovation in supply chains*. *Journal of Digital Economy* , 4, 108–122. <https://doi.org/10.1016/j.jdec.2025.08.002>
- [3] du Plessis, MJ, Gerber, R., Goedhals-Gerber, LL, & van Eeden, J. (2025). *Shaping the future of freight logistics: Use cases of artificial intelligence*. *Sustainability* , 17(4), 1355. <https://doi.org/10.3390/su17041355>
- [4] Oletsy, O., & Moholivskyi, V. (2024). *On supervising and coordinating microservices within web applications on the basis of state machines*. *CEUR Workshop Proceedings* , 3909, 442–454.
- [5] Miriyala, NS (2025). *Study of orchestration engines: Open-source workflow & cloud-native solutions*. *Stochastic Modelling and Computational Sciences* , 5(1), 1–16.
- [6] Han, W. (2025, November). *An Empirical Study on Building a High-precision Forecasting System for Financial Market Trends Based on the PyTorch Deep Learning Framework*. In

- International Conference on Computational Technologies for Research in Data Analytics* (pp. 518-528). Cham: Springer Nature Switzerland.
- [7] Silva, MVS, Santos, LFC, Soares, MS, & Rocha, FG (2025). *Guidelines for the application of event driven architecture in micro services with high volume of data*. In *Proceedings of the 27th International Conference on Enterprise Information Systems* , 859–866. <https://doi.org/10.5220/0013348600003929>
- [8] Guo, X. (2025, April). *Research on Financial Trading Algorithms Based on Deep Reinforcement Learning*. In *2025 IEEE 3rd International Conference on Control, Electronics and Computer Technology (ICCECT)* (pp. 1711-1715). IEEE.
- [9] Guo, X. (2025, March). *Research on Blockchain-Based Financial AI Algorithm Integration Methods and Systems*. In *2025 IEEE International Conference on Electronics, Energy Systems and Power Engineering (EESPE)* (pp. 816-821). IEEE.
- [10] Jing, X. (2025, November). *In-Depth Analysis of the Optimization Decision Model of Financial Investment Portfolio Based on Machine Learning Algorithms in Python Scikit-Learn Library*. In *International Conference on Computational Technologies for Research in Data Analytics* (pp. 130-140). Cham: Springer Nature Switzerland.