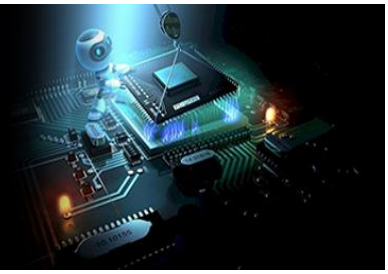
International Journal of Engineering in Computer Science



E-ISSN: 2663-3590
P-ISSN: 2663-3582
www.computersciencejournals.com/ijecs
IJECS 2025; 7(2): 12-17
Received: 17-04-2025
Accepted: 21-05-2025

Mykyta Roilian
Engineering Manager, LeanDNA, 611 S Congress Ave Ste 300, Austin, Texas 78704, USA

Corresponding Author:
Mykyta Roilian
Engineering Manager, LeanDNA, 611 S Congress Ave Ste 300, Austin, Texas 78704, USA

Designing fault-tolerant distributed systems for supply chain management: Architectural patterns and manufacturing scenarios

Mykyta Roilian

DOI: https://www.doi.org/10.33545/26633582.2025.v7.i2a.193

Abstract
This article presents an analysis of architectural solutions aimed at ensuring the fault tolerance of distributed information systems used in supply chain management within industrial environments. Three primary patterns are examined: replication, fallback mechanisms, and graceful degradation. The study evaluates their applicability across various failure scenarios and provides a comparative assessment based on recovery time objectives (RTO and MTTR) and system availability. The necessity of integrated architectural design is emphasized, considering the criticality of business functions. Recommendations are proposed for the implementation of fault-tolerant strategies at different infrastructure levels to improve the reliability and predictability of supply chain management systems.

Keywords: Distributed systems, fault tolerance, replication, fallback, graceful degradation, supply chain management, industrial IT infrastructure

1. Introduction
The digitalization of manufacturing processes requires enterprises to maintain a high level of IT system resilience, particularly in the context of supply management, where continuous availability of services tracking procurement, material movement, and consumption is critical. In high-tech industries, a failure in the supply information system can disrupt assembly schedules, cause production line downtime, and result in significant financial losses. Unlike traditional supply management systems in manufacturing are focused on maintaining an exact balance between the needs of the assembly line and current inventory levels. This imposes strict requirements on the architecture of such systems to ensure robustness against failures - both within the internal infrastructure and across external services, including supplier interfaces and corporate ERP platforms.
The objective of this study is to analyze architectural patterns used in the design of fault-tolerant distributed systems applicable to supply management tasks in manufacturing enterprises.

2. Main part. Architectural patterns of fault tolerance
The design of fault-tolerant distributed systems requires the implementation of architectural solutions that ensure the uninterrupted operation of critical components in the event of partial or complete infrastructure failures. In a manufacturing environment, where inventory and supply management must remain both available and consistent regardless of disruptions, specific fault-tolerance patterns are applied. The most widely used among them - replication, fallback mechanisms, and graceful degradation - are examined below, with emphasis on their practical applicability in supply chain management tasks.
Data and service replication is a method of failure protection that involves creating multiple instances of system components distributed across different failure zones (fig. 1).

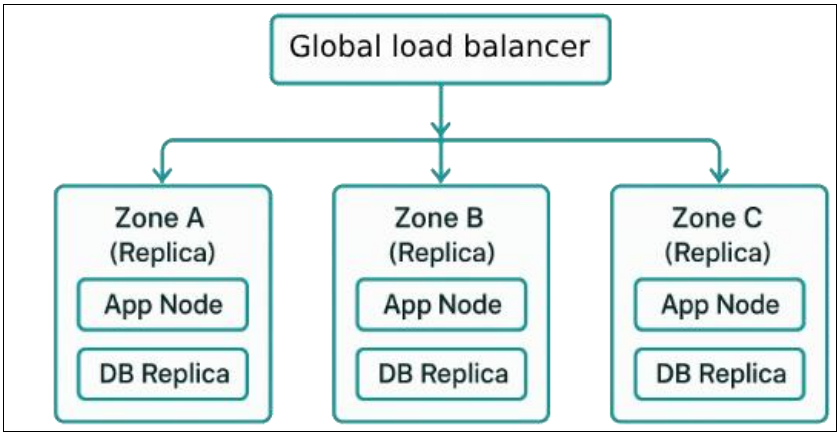


Fig 1: Data replication scheme.

Depending on consistency and response time requirements, either synchronous or asynchronous replication is used. Synchronous replication ensures data coherence but increases write latency and demands stable communication. Asynchronous replication eases load on the primary node but may cause data inconsistencies during failures.

In supply chain systems, synchronous replication is preferred for critical data such as material specifications and order statuses. According to the State of Logistics Survey 2024 [1], top investment areas include digitalization (21, 5%), data analytics (18, 4%), and real-time visibility (15, 6%), all requiring fast and reliable data exchange.

Companies like FedEx, Bolloré Logistics, and LKW Walter prioritize transparency and fault tolerance, adopting distributed IT models with real-time data replication to support these goals.

In addition to replication, fallback mechanisms are widely used to enable load switching from failed components to backup or alternative ones. This approach allows transactions to continue without interruption in case of unavailability of an external API, internal service, or database (fig.2).

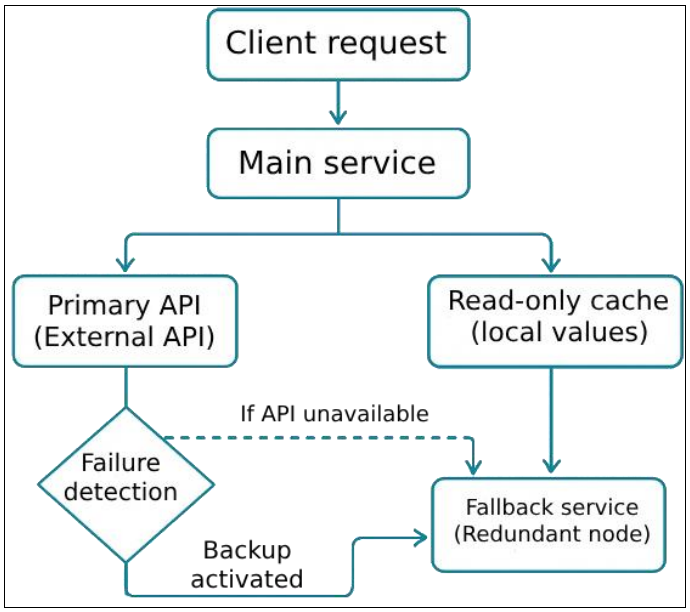


Fig 2: Fallback mechanism in service resilience architecture.

In industrial IT systems, fallback mechanisms are implemented using active-passive or active-active configurations. In active-passive, the secondary node activates after a failure, typically with a 10-30 second delay. Active-active runs multiple nodes in parallel, enabling near-instant failover. For external services like supplier APIs, fallback often uses read-only caches with pre-approved values valid for up to 24 hours, triggered by errors (e.g., HTTP 4xx/5xx). These mechanisms require automated monitoring, request routing, and event-driven orchestration to quickly redirect operations and maintain functionality during component failures.

The third approach - graceful degradation - refers to the system's ability to continue operating despite the degradation of certain functions or services, ensuring that core business logic remains intact (fig. 3).

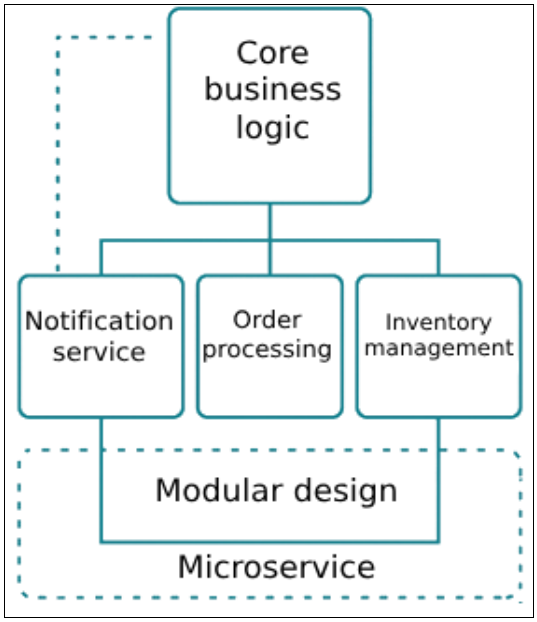


Fig 3: Modular microservice architecture supporting graceful degradation of non-critical functions.

This approach is most useful during overloads or failures of non-critical components. For instance, if a database is partially unavailable, the system can temporarily disable analytics while keeping core functions like order processing active. It may also slow external data sync or limit the interface to essential operations [2]. To support this, systems should follow a modular, microservice-based design with scalable, independently deactivatable components and clear functional priorities. Together, fault-tolerant patterns provide a reliable strategy for maintaining supply chain continuity, minimizing downtime, and ensuring predictable behavior during failures.

3. Application in Manufacturing Scenarios

Industrial enterprises dealing with a large number of material items, multiple suppliers, and distributed warehouse infrastructures place high demands on the resilience of IT systems that support supply chain management. In the context of complex, multi-stage production cycles, the failure of a single component within the digital platform can lead to a breakdown in synchronization between production planning, inventory management, and external supply sources. Practical experience shows that the effective implementation of fault-tolerant architectures depends on the type of manufacturing process, the criticality of supplies, and the specifics of supplier interaction (table 1).

Table 1: Application of fault-tolerance patterns in manufacturing supply chain scenarios [3, 4]

Manufacturing process scenario	Type of failure	Applied pattern	Implementation	Outcome
Failure of the central order management module	Database service failure	Replication (synchronous)	Horizontally scalable storage with a backup node.	Preservation of transactional consistency.
Unavailability of external supplier API	External connection failure	Fallback	Use of locally cached data or duplicated channels.	Minimization of decision-making delays.
Overload of material demand calculation service	Performance degradation	Graceful degradation	Lower frequency, core functions only.	Retention of core functionality.
Failure of inter-site communication service	Network failure	Fallback + replication	Local storage of orders with post-recovery consolidation	Continuity of operations at isolated sites.
Inaccessibility of consumption analytics module	Auxiliary service failure	Graceful degradation	Prioritized core functions.	Maintenance of critical-level functional capability.
Planned overload of ERP system nodes	Planned unavailability	Replication + Fallback	Switching to backup instance with full synchronization.	Elimination of process downtime during operational hours.

Modern industrial IT systems not only implement basic fault-tolerance mechanisms but are increasingly integrating intelligent analytics modules to enhance adaptability. For example, the application of artificial intelligence in warehouse operations contributes to greater resilience against unexpected disruptions and enables real-time optimization of supply chain decisions. AI technologies are being actively applied in warehouse operations management to improve resource allocation, forecast demand more accurately, and reduce manual errors in high-load environments [5].

The scenarios presented illustrate how technical architectural solutions can be embedded into the context of real-world manufacturing processes. The application of each pattern is determined by the criticality level of the supported function: for order and inventory management, it typically involves replication with full data coherence; for interactions with external systems - fallback mechanisms with acceptable latency; and for analytics and support - degradation strategies allowing temporary deactivation. An approach focused on isolation, scalability, and component resilience helps minimize the impact of local failures on the entire supply chain. Thus, architectural patterns become an integral part of designing the information infrastructure of a modern manufacturing enterprise.

4. Comparative analysis of pattern effectiveness

The choice of a fault-tolerant architectural pattern should be

based on key technical metrics such as Recovery Time Objective (RTO), Mean Time to Recovery (MTTR), and system availability. This section presents a comparative analysis of three patterns - replication, fallback, and graceful degradation - using five typical failure scenarios in manufacturing environments to objectively assess the resilience of each architecture under real-world conditions.

The first metric - RTO - represents the maximum acceptable time for system recovery after a failure. The replication pattern provides the lowest RTO, especially in active node configurations, where failover occurs almost instantly. This is crucial for maintaining synchronous operations in production environments that demand minimal downtime. Fallback mechanisms yield moderate RTO values, as they require time to detect failures and reroute requests. The highest RTO is observed in the graceful degradation pattern, where service interruptions are mitigated by temporarily reducing functionality, lowering the urgency of immediate recovery (fig. 4).

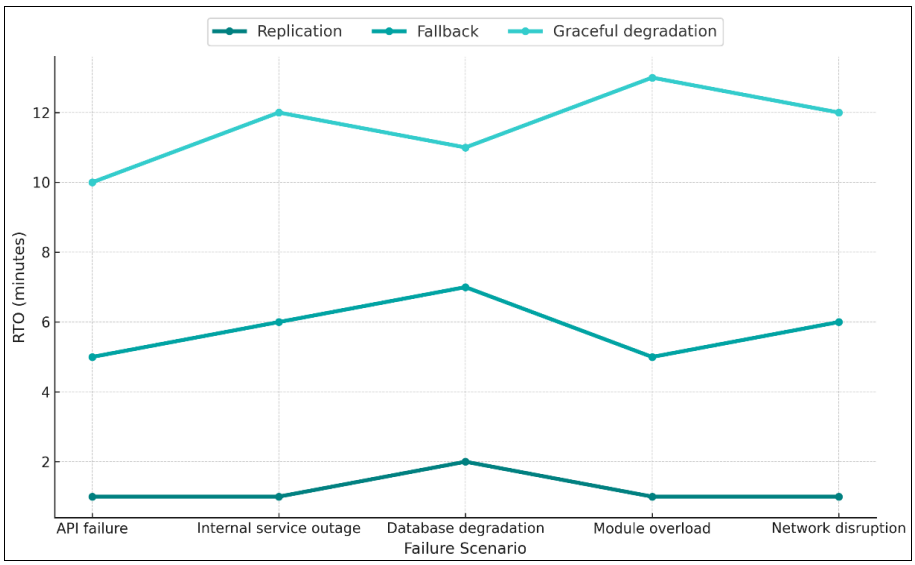


Fig 4: RTO across failure scenarios for fault-tolerance patterns.

The RTO values presented in the graph are based on engineering best practices and industry sources that reflect the behavior of common fault-tolerant patterns in production IT environments [6, 7]. According to AWS, the active replication strategy (multi-site active-active) delivers the lowest RTO and RPO, as failure triggers immediate traffic redirection across regions without node shutdown - effectively with near-zero delay. Overall, minimal RTO values for replication (1-2 minutes) correspond to active configurations with automated failover, as confirmed by documentation on high-availability clusters and cloud systems. For the fallback approach (5-7 minutes), delays are typically caused by failure detection and rerouting logic. Higher RTO values for graceful degradation (10-13 minutes) result from the architecture's focus on maintaining core functionality under load rather than achieving rapid recovery.

The second key metric is system availability, expressed as the percentage of time the system remains operational. Replication demonstrates the highest level of stability, as the failure of a single component does not lead to service interruption. Fallback mechanisms provide a high, though slightly less consistent, level of availability - especially when external dependencies are involved. Graceful degradation maintains overall operability but, due to reduced functionality, the system's full availability is assessed as lower (fig. 5).

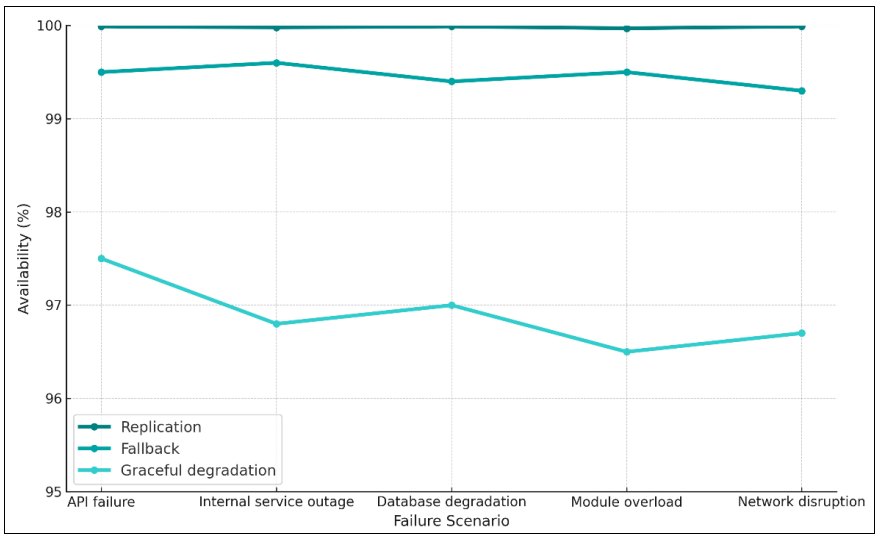


Fig 5: System availability by resilience pattern under various failure scenarios.

For the replication pattern, the availability level in the range of 99, 97-99, 99% reflects the characteristics of high-availability solutions, where automatic failover and load balancing ensure virtually uninterrupted service operation [8, 9]. The availability level for fallback mechanisms (99,3-99,6%) is determined by the time required for failure detection and traffic rerouting, as well as the potential dependency on external API, whose SLA rarely exceed 99,5%. In the case of graceful degradation, the reduction in full availability to 96, 5-97, 5% is due to the fact that the system retains only its core functionality, temporarily limiting auxiliary services such as analytics, synchronization, or interface modules.

The third metric - MTTR- reflects the average time required to restore the system after a failure and provides insight into the system's typical response to incidents. This value captures not only the technical characteristics of the infrastructure but also the efficiency of monitoring, alerting, and automated response processes, determining how quickly the system can return to a stable state after a disruption (fig. 6).

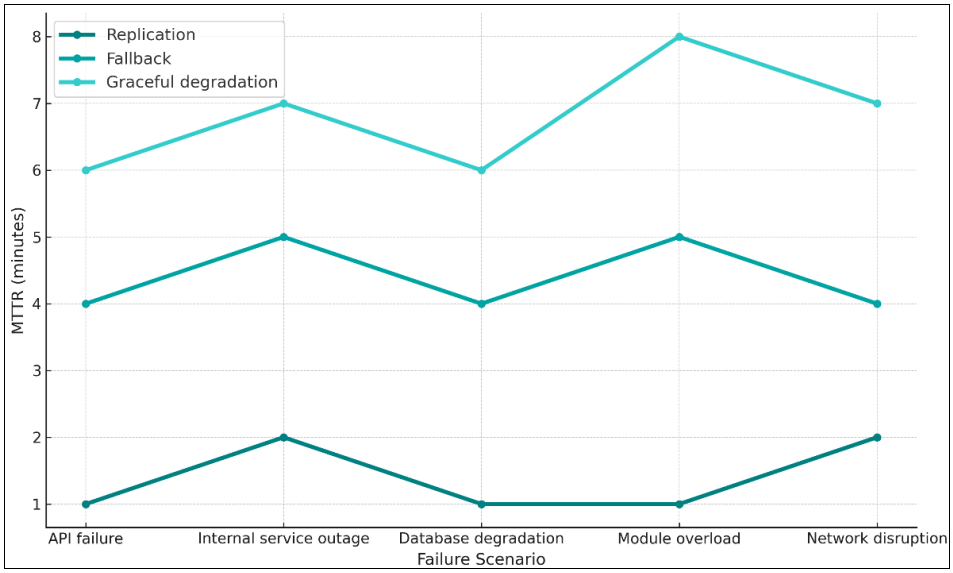


Fig 6: Comparative MTTR values across failure scenarios for different resilience patterns.

For replication, the low MTTR value (1-2 minutes) is explained by the presence of a pre-configured standby node and automatic failover without the need for initialization. In the case of fallback mechanisms, the average recovery time increases to 4-5 minutes due to the need to identify the failure point, reroute traffic, and possibly load alternative components. Graceful degradation demonstrates the highest MTTR (6-8 minutes), as it often requires manual or operator-assisted recovery of disabled functions, along with re-synchronization of modules [10].

Overall, replication is considered the most suitable solution for critical systems with strict continuity requirements. Fallback handles external failures effectively and can be efficiently implemented with caching and redundant interfaces. Despite its relatively high recovery times, graceful degradation enables the system to maintain essential functionality without complete shutdown, which is particularly valuable in user-facing or interactive environments. In industrial practice, the best results are achieved through the combined integration of all three approaches, distributed according to the criticality levels of business functions.

5. Conclusion

Given the complexity of modern production chains and the reliance on stable information flows, fault tolerance in distributed supply chain management systems is a critical requirement. Architectural patterns - replication, fallback, and graceful degradation - offer strategies to mitigate various types of failures and maintain operational continuity. Replication provides maximum availability and minimal recovery time by duplicating critical components. Fallback allows for flexible responses to external service failures through alternative channels or cached data. Graceful degradation mechanisms, in turn, enable the preservation of core functionality even when some services become unavailable, which is particularly important for user-facing interfaces and operational continuity.

Based on the conducted analysis, several recommendations can be made for designing fault-tolerant supply chain systems. First, a multi-layered approach should be implemented, combining different architectural patterns according to the criticality of business functions. Second, services should be decomposed into distinct failure domains, allowing for independent scaling and monitoring. Third, systems should incorporate prioritized degradation mechanisms, ensuring that essential operations continue under constrained conditions. Finally, fault tolerance should be viewed as an integral property of the system architecture, rather than as a set of auxiliary features added in the later stages of development. This approach not only increases system reliability but also improves the long-term manageability of digital infrastructure.

6. References

1. Transport Intelligence. State of logistics survey 2024: road freight - key challenges and technology investments. 2024. https://www.ti-insight.com/wp-content/uploads/2024/04/Road-Transport-SOLS-2024-WP-3.pdf (accessed 2025 Jun 23).
2. Allam H. Full-stack resilience: Designing systems that tolerate chaos by default. Int J Emerg Res Eng Technol.

2025;6(2):53-62.
3. Lijin Z. Research on software supply chain security assurance mechanism system based on blockchain technology. In: 2024 4th International Signal Processing, Communications and Engineering Management Conference (ISPCEM); 2024. p. 939-944.
4. Stepanov M. Implementation of CRM and ERP systems in small and medium-sized businesses as a means of improving operational efficiency in e-commerce. Econ Bus Theory Pract. 2025;5(123):382-386.
5. Shokirov K. Application of artificial intelligence in warehouse operations management. Prof Bull Econ Manag. 2024;4:29-33.
6. Amazon Web Services. REL13-BP02: Use defined recovery strategies to meet the recovery objectives. 2022. https://docs.aws.amazon.com/wellarchitected/2022-03-31/framework/rel_planning_for_recovery_disaster_recovery.html (accessed 2025 Jun 24).
7. Amazon Web Services. Disaster recovery of workloads on AWS: Recovery in the cloud. 2022. https://docs.aws.amazon.com/pdfs/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-workloads-on-aws.pdf (accessed 2025 Jun 24).
8. Amazon. Amazon Aurora Service Level Agreement. https://aws.amazon.com/ru/rds/aurora/sla/ (accessed 2025 Jun 24).
9. Amazon. Amazon Elastic Load Balancing Service Level Agreement. https://aws.amazon.com/ru/elasticloadbalancing/sla/ (accessed 2025 Jun 24).
10. Rahman F, Soewito B. Enhancing database availability: A combined approach using SQL Always On Failover Cluster Instance and Availability Groups. J Comput Sci. 2025;21(6):1332-1342.