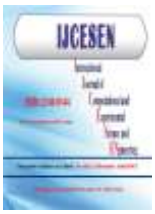
International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 12-No.1 (2026) pp. 59-66
http://www.ijcesen.com



ISSN: 2149-9144

Research Article

Self-Healing RPA Systems: A Reliability-Centric Architecture for Financial Enterprises

Ravindra Reddy Madireddy*

Independent Researcher, USA
* **Corresponding Author Email:** madireddy.ravindrareddy@gmail.com - **ORCID:** 0000-0002-5007-7811

Article Info:

DOI: 10.22399/ijcesen.4622
Received : 29 November 2025
Revised : 01 January 2026
Accepted : 02 January 2026

Keywords

Self-Healing Systems,
Robotic Process Automation,
Financial Operations Automation,
Fault-Tolerant Computing,
Autonomous Recovery Mechanisms,
Reliability Engineering

Abstract:

Financial institutions are struggling to keep their robotic process automation reliable, even though their operational environments are constantly changing. Traditional RPA deployments are very fragile and, as a result, are affected by changes in the interface, variations in the infrastructure, and even changes in the data formats. Manual intervention requirements undermine automation value propositions by extending recovery timelines and consuming support resources. The self-healing architecture presented addresses fundamental limitations through integrated telemetry capture, intelligent diagnostics, adaptive recovery mechanisms, and continuous learning capabilities. Multi-layer designs embed autonomous corrective logic directly within RPA execution frameworks rather than relying on external monitoring systems. Real-time telemetry streams enable pattern recognition algorithms to classify failure types and route incidents to appropriate remediation procedures. Adaptive selector management maintains hierarchical fallback chains spanning multiple element identification strategies. Exception routing logic distinguishes transient faults amenable to automated recovery from structural defects requiring human expertise. Financial operations implementations demonstrate practical applications across payment processing, account reconciliation, and regulatory validation workflows. Reliability engineering principles establish observability frameworks, measuring recovery efficacy and diagnostic accuracy. Continual learning architectures refine classification models through feedback loops, capturing production outcomes. The architectural framework transforms automation reliability from static design properties into dynamic operational capabilities evolving alongside environmental changes.

1. Introduction

Financial operations processing environments demand continuous workflow execution across payment gateways, ledger systems, compliance platforms, and data warehouses. Robotic process automation has emerged as the standard mechanism for orchestrating rule-driven activities that previously consumed substantial human resources. However, the inherent brittleness of conventional RPA implementations creates operational vulnerabilities. Interface element identifiers become obsolete following application updates. Database query timeouts disrupt transaction fetches. Unexpectedly, data structures cause unhandled exceptions. These failure modes accumulate over time, which leads to automation reliability degradation and thus necessitates the presence of dedicated support teams for issue diagnosis and manual process restarts.

Multi-system integration presents substantial challenges for RPA deployments in financial institutions. Workflow automation must coordinate activities across heterogeneous application architectures with varying response characteristics and data formats [1]. Legacy systems introduce additional complexity through inconsistent API behaviors and unpredictable error responses. Application updates occur without synchronized notification to automation teams. Interface modifications break existing selector logic. Data schema evolution renders transformation rules obsolete. The cumulative effect manifests as progressive reliability degradation that undermines automation value propositions.

Current RPA architectures operate under reactive maintenance paradigms where failures trigger alerts

that queue for human review. Support engineers must examine log files, reproduce error conditions, and deploy corrective patches. This process extends the mean time to recovery and reduces effective automation uptime. The fundamental limitation lies in the absence of autonomous corrective capabilities within the RPA runtime environment itself. Existing orchestration platforms provide monitoring dashboards and exception logging, but lack integrated intelligence to interpret failure signals and execute remediation procedures dynamically. Manual intervention requirements constrain the scalability potential for enterprise automation programs [1].

Machine learning-powered self-healing mechanisms offer architectural alternatives to conventional reactive maintenance models. Automated defect detection algorithms analyze execution telemetry to identify anomalous behaviors before complete workflow failures occur [2]. Predictive maintenance capabilities enable proactive remediation of degrading system conditions. Pattern recognition techniques classify failure types and route incidents to appropriate recovery procedures. The integration of intelligent diagnostic logic directly into automation runtimes transforms failure handling from manual processes into autonomous system capabilities [2].

This paper addresses the reliability gap by presenting a self-healing architecture that embeds diagnostic intelligence and automated recovery logic directly into the RPA execution framework. The contribution encompasses a multi-layer design spanning telemetry capture, anomaly detection, decision routing, and action execution. By treating operational failures as recoverable events rather than terminal errors, the architecture enables financial automation systems to sustain workflow continuity through environmental disruptions. The following sections detail the architectural components, recovery mechanisms, implementation patterns for financial processes, and reliability engineering practices that collectively enable autonomous healing capabilities.

2. Related Work and Methodology

Existing literature on RPA reliability predominantly addresses reactive monitoring approaches where human operators respond to failure alerts. Traditional frameworks lack integrated autonomous recovery capabilities within runtime environments. Recent advances in fault-tolerant distributed systems provide foundational principles for transaction consistency and atomic commitment protocols applicable to financial automation contexts. Machine learning applications in industrial fault detection demonstrate supervised and unsupervised techniques for anomaly classification. However, prior efforts focus primarily on detection rather than automated remediation execution.

The methodology introduces a multi-layer architecture integrating four distinct functional domains. The telemetry layer captures comprehensive execution traces spanning bot runtimes, target applications, and infrastructure components. Real-time processing pipelines apply pattern recognition algorithms to streaming event data. The diagnostic engine employs decision trees and ensemble classifiers trained on historical failure repositories to categorize disruptions and recommend resolution strategies. Adaptive recovery mechanisms implement hierarchical selector fallback chains combining XPath expressions, CSS selectors, positional logic, and computer vision techniques. Exception routing logic applies recoverability criteria, distinguishing transient faults from structural defects. Continual learning frameworks incorporate production outcomes into expanding training datasets through experience replay and progressive neural architectures. Financial operations validation demonstrates practical implementations across payment processing, reconciliation workflows, and regulatory validation contexts. Observability infrastructure measures autonomy levels through behavioral analysis rather than architectural assumptions. The framework's primary contribution lies in embedding autonomous diagnostic intelligence and recovery execution directly within RPA runtimes.

3. Architectural Framework for Self-Healing RPA

3.1 Telemetry and Event Capture Layer

The foundation of self-healing capabilities rests on comprehensive telemetry collection across all automation touchpoints. Every bot execution generates temporal event streams capturing selector resolution outcomes, response time distributions, exception classifications, and transaction state transitions. The telemetry layer instruments not only the RPA runtime but also target applications, middleware components, and data sources involved in end-to-end workflows.

Real-time monitoring frameworks leverage artificial intelligence algorithms to process continuous data streams from distributed automation environments. Convolutional neural networks analyze temporal patterns in execution telemetry to identify anomalous behaviors

indicative of emerging failures [3]. Recurrent architectures maintain memory of historical execution states to detect gradual performance degradation. Deep learning models extract features from raw telemetry without requiring manual pattern specification. Edge computing deployments enable local analysis to reduce latency in failure detection workflows [3].

Event schemas provide a framework for storing metadata such as process identifiers, step sequences, resource consumption metrics, and environmental context, like queue depths and system loads. Time series forecasting models are used to predict future resource requirements on the basis of past utilization trends. Anomaly detection algorithms are used to identify deviations from the expected operational baselines [3]. This instrumentation enables correlation analysis between infrastructure conditions and failure patterns, establishing causality rather than mere temporal associations. Predictive analytics capabilities transform reactive monitoring into proactive failure prevention strategies.

3.2 Diagnostic Engine

The diagnostic engine consumes telemetry streams through real-time processing pipelines that apply pattern matching, statistical deviation detection, and sequence analysis algorithms. When an automation step fails, the engine examines recent event history to classify the failure type. Selector mismatches indicate UI changes. Timeouts suggest downstream latency issues. Data validation rejections point to format inconsistencies. Authorization failures signal credential expiration.

Machine learning approaches address fault detection and diagnosis challenges inherent in complex automation ecosystems. Supervised learning algorithms require labeled training datasets containing historical failure instances with verified root cause annotations [4]. Classification models distinguish between different failure categories based on telemetry feature vectors. Support vector machines establish decision boundaries separating distinct fault types in high-dimensional feature spaces. Random forest ensembles combine multiple decision trees to improve classification robustness against noisy input data [4].

Unsupervised learning techniques identify novel failure patterns absent from historical training repositories. Clustering algorithms group similar failure signatures without prior category definitions. Dimensionality reduction methods project high-dimensional telemetry into interpretable feature spaces [4]. Semi-supervised approaches leverage small labeled datasets augmented with abundant unlabeled execution logs to improve model generalization. The diagnostic output includes not only failure categorization but also confidence scores and recommended remediation strategies ranked by success probability based on past recovery outcomes.

Deep neural architectures process sequential telemetry data to capture temporal dependencies in failure evolution patterns. Long short-term memory networks maintain context across extended execution traces [4]. Attention mechanisms focus diagnostic analysis on the most informative telemetry segments. Transfer learning adapts models trained on general fault detection tasks to specific financial automation contexts with limited domain-specific training data.

4. Intelligent Recovery Mechanisms

4.1 Adaptive Selector Management

Interface element identification represents a primary fragility vector in RPA systems. Applications undergo frequent updates that modify DOM structures, CSS class assignments, and accessibility properties without notification to automation maintainers. The adaptive selector management subsystem addresses this challenge through multi-strategy element location. Rather than relying on single identifiers, the recovery mechanism maintains selector hierarchies with fallback chains spanning XPath expressions, CSS selectors, relative positioning logic, and OCR-based text recognition.

Similarity-based localization techniques address element identification challenges when traditional selector strategies fail. Visual similarity metrics compare current interface states against reference screenshots to identify target elements despite structural changes [5]. Textual content matching algorithms locate elements through semantic analysis of visible text labels and surrounding context. Structural similarity computation analyzes DOM tree patterns to find elements occupying equivalent positions in modified interface hierarchies [5].

When the primary selector fails, the system systematically attempts alternatives while validating element properties against expected characteristics. Hierarchical fallback mechanisms progress through increasingly flexible matching criteria. Exact attribute matches provide the highest confidence identification. Partial attribute matching tolerates minor variations in element properties. Positional similarity considers spatial relationships between interface components [5]. Successful recoveries update the selector preference ordering,

creating learning loops that optimize future identification attempts. Machine learning models learn from successful recovery patterns to predict optimal selector strategies for specific application contexts.

4.2 Exception Routing Logic

Not all failures warrant automated recovery attempts. The routing subsystem evaluates diagnostic outputs against predefined recoverability criteria to determine appropriate response pathways. Exception intelligence frameworks classify failures into distinct typologies based on root cause characteristics and resolution requirements [6]. Transient network failures trigger exponential backoff retry sequences with jitter to avoid synchronized load spikes. Data format mismatches invoke transformation routines that normalize inputs according to schema definitions maintained in the configuration repository.

Playbook-based resolution systems codify recovery procedures for recurring failure patterns encountered in production environments. Structured playbooks document diagnostic steps, remediation actions, and escalation criteria for each exception category [6]. Real-time resolution engines execute playbook instructions automatically when exception patterns match predefined signatures. Decision logic evaluates contextual factors, including failure frequency, business impact severity, and available recovery resources, to select appropriate response strategies [6].

Authentication errors activate credential refresh procedures that interact with secrets management infrastructure. Structural failures indicating application logic changes bypass automated remediation and route incident tickets to development teams with contextual diagnostic information. Intelligent routing distinguishes between exceptions amenable to autonomous resolution and those requiring human expertise [6]. This classification prevents futile recovery loops and ensures resources focus on genuinely resolvable conditions. Feedback mechanisms capture resolution outcomes to refine exception typologies and improve routing accuracy over time.

5. Implementation in Financial Operations

Transaction Processing Workflows

Payment processing automation frequently encounters failures during gateway interactions, clearing system communications, and ledger updates. Self-healing architectures embed recovery logic at each integration boundary. When payment gateway APIs return timeout errors, the system verifies transaction status through reconciliation endpoints before attempting resubmission, preventing duplicate charges.

Fault-tolerant distributed system principles govern transaction processing reliability in financial automation environments. Atomic commitment protocols ensure transaction consistency across multiple participating systems [7]. Partial failures require coordinated recovery mechanisms that maintain system-wide coherence. Byzantine failure modes complicate recovery logic when subsystems exhibit arbitrary incorrect behaviors rather than simple crash failures [7]. Recovery algorithms must distinguish between permanent component failures requiring failover and transient faults amenable to retry strategies.

Clearing system connectivity losses activates queue persistence mechanisms that buffer transactions locally until network paths are restored, maintaining processing continuity without data loss. Ledger update failures trigger compensating transaction protocols that reverse partial entries and re-execute complete transaction sequences atomically. Checkpointing strategies capture consistent global states, enabling rollback to known-good configurations following cascading failures [7]. Message ordering guarantees prevent transaction sequence violations during recovery operations. Idempotency enforcement ensures duplicate message delivery does not compromise financial accuracy.

5.1 Reconciliation and Validation Processes

Automated account reconciliation should be capable of managing situations where the data source is unavailable, changes in schema, and variations in calculation logic. Recovery mechanisms cache reference data with staleness awareness, enabling reconciliation to proceed using recently validated snapshots when source systems become unreachable. Data-centric architectural patterns establish unified data foundations supporting reconciliation workflows across heterogeneous financial systems [8].

Platform integration challenges arise from disparate data models, inconsistent semantics, and varying update frequencies across enterprise financial applications. Modern architectures implement abstraction layers that normalize data representations and provide consistent access interfaces regardless of underlying source system characteristics [8]. Real-time data pipelines enable continuous reconciliation rather than batch-oriented overnight processes. Event-driven architectures propagate financial state changes immediately to dependent systems.

Schema mismatches invoke adaptive parsers that attempt field mapping through semantic analysis of column headers and data patterns. Data governance frameworks establish canonical data models defining standard representations for financial entities across the enterprise [8]. Metadata registries document schema relationships and transformation logic connecting source systems to consolidated views. When calculation discrepancies exceed tolerance thresholds, diagnostic routines compare algorithm versions between reconciliation bots and source systems, flagging logic drift for manual review while isolating affected accounts to prevent batch failures.

Integration platforms provide orchestration capabilities, coordinating data flows between transaction processing systems, reconciliation engines, and reporting applications [8]. API management layers standardize access patterns and enforce consistency requirements. The shift in systems layout from standalone programs to interconnected information-centric systems considerably improves reconciliation reliability as it reduces the complexity of point-to-factor integrations.

6. Reliability Engineering and Continuous Optimization

6.1 Monitoring Infrastructure

Sustained self-healing capability requires observability infrastructure that exposes not only traditional uptime metrics but also recovery success rates, failure classification accuracy, and remediation latency distributions. Monitoring dashboards visualize healing efficacy through time series displaying manual intervention frequency, automated recovery attempt outcomes, and residual failure categories.

Measuring autonomy levels in self-healing systems requires objective frameworks based on observable behavioral characteristics rather than architectural assumptions. Autonomy assessment methodologies evaluate system capabilities across multiple dimensions, including decision-making independence, environmental adaptation, and self-maintenance proficiency [9]. Observable behaviors provide empirical evidence of autonomous operation. Action selection without human guidance demonstrates decision autonomy. Environmental response patterns reveal adaptation capabilities. Self-correction behaviors indicate maintenance autonomy [9].

Alert thresholds trigger when recovery success rates degrade, indicating either environmental changes requiring selector updates or new failure modes absent from diagnostic training data. Behavioral monitoring captures system responses to failure conditions across temporal sequences. Autonomy metrics quantify the degree to which systems operate independently during disruption scenarios [9]. Performance degradation patterns signal when autonomous capabilities require enhancement through additional training data or refined decision logic.

6.2 Iterative Learning Framework

The architecture incorporates feedback loops that capture recovery outcome data for continuous model refinement. Successful automated remediations update precedent libraries that inform future diagnostic recommendations. Recovery failures generate training samples that enhance classification models and expand remediation strategy repositories. Continual learning frameworks address the challenge of maintaining model performance as operational environments evolve [10].

Industrial applications confront concept drift phenomena where data distributions shift gradually due to changing business processes, system upgrades, and external market conditions. Static models trained on historical data exhibit degrading accuracy when confronted with novel patterns absent from original training sets [10]. Catastrophic forgetting occurs when neural networks lose previously acquired knowledge upon training with new data samples. Continual learning architectures mitigate this limitation through memory replay mechanisms and dynamic network expansion strategies [10].

Periodic analysis identifies failure pattern clusters that suggest systemic issues requiring architectural modifications rather than tactical recovery logic. This iterative improvement cycle ensures self-healing capabilities evolve in tandem with operational environment changes. Experience replay buffers maintain representative samples from historical failure distributions to prevent knowledge erosion during model updates [10]. Elastic weight consolidation techniques protect important neural network parameters corresponding to previously learned failure patterns. Progressive neural architectures dynamically allocate additional capacity for emerging failure categories while preserving existing classification capabilities [10]. Task-specific adaptation layers enable specialized handling of distinct failure contexts without requiring complete model retraining.

Table 1. *Architectural Components of Self-Healing RPA Framework: Telemetry Capture and Diagnostic Processing Elements [3, 4].*

Component	Function	Key Capabilities
Telemetry Layer	Event stream capture	Selector resolution tracking, response time monitoring, exception classification, transaction state logging
Event Schema	Metadata management	Process identifiers, step sequences, resource metrics, queue depths, system loads
Diagnostic Engine	Failure classification	Pattern matching, statistical deviation detection, sequence analysis, confidence scoring
Processing Pipeline	Real-time analysis	Convolutional neural networks for temporal patterns, recurrent architectures for historical states
Classification Logic	Root cause determination	Decision trees distinguishing selector mismatches, timeouts, validation failures, and authorization errors

Table 2. *Recovery Mechanism Strategies and Implementation Approaches, Adaptive Selector Management and Exception Routing Frameworks [5, 6].*

Recovery Strategy	Application Context	Operational Characteristics
Hierarchical Selectors	Interface element identification	XPath expressions, CSS selectors, relative positioning, and OCR-based recognition
Similarity-Based Localization	Dynamic interface adaptation	Visual similarity metrics, textual content matching, and structural similarity computation
Exponential Backoff	Transient network failures	Progressive retry intervals with random jitter, preventing synchronized load spikes
Schema Normalization	Data format mismatches	Transformation routines supporting regional formats, encoding variations, and currency symbols
Credential Refresh	Authentication failures	Token renewal workflows interacting with secrets management infrastructure
Incident Escalation	Structural defects	Automated ticket generation with diagnostic context for development teams

Table 3. *Financial Operations Implementation Patterns Transaction Processing and Reconciliation Automation Scenarios [7, 8].*

Financial Process	Failure Scenarios	Self-Healing Response
Payment Gateway Integration	Timeout errors, connectivity loss	Transaction status verification, duplicate charge prevention, and idempotency enforcement
Clearing System Communication	Network disruptions	Queue persistence with durable storage, circuit breaker patterns, and controlled replay mechanisms
Ledger Updates	Partial transaction failures	Compensating transaction protocols, two-phase commit coordination, saga pattern implementation
Account Reconciliation	Data source unavailability	Reference data caching with staleness tracking, time-to-live metadata management
Schema Evolution Handling	Field mapping inconsistencies	Natural language processing for column analysis, type inference from sample values
Calculation Validation	Algorithm version drift	Version control integration, discrepancy threshold monitoring, and affected account isolation

Table 4. *Continuous Optimization Framework Components, Monitoring Infrastructure, and Iterative Learning Mechanisms [9, 10].*

Framework Element	Measurement Focus	Optimization Technique
Observability Infrastructure	Recovery success rates, classification accuracy	Multi-threshold escalation policies, anomaly detection algorithms, and service level tracking
Behavioral Monitoring	Autonomy level assessment	Decision-making independence, environmental adaptation, and self-maintenance proficiency
Feedback Loops	Recovery outcome capture	Precedent library updates, training sample generation, and model refinement cycles
Continual Learning	Concept drift mitigation	Experience replay buffers, elastic weight consolidation, progressive neural architectures
Pattern Clustering	Systemic issue identification	Failure mode taxonomy expansion, architectural modification triggers

Transfer Learning	Domain adaptation	Model reuse across automation contexts, accelerated capability development

7. Conclusions

The architectural framework presented establishes practical pathways for financial institutions seeking to enhance automation reliability through autonomous healing capabilities. Traditional RPA deployments remain vulnerable to environmental perturbations absent integrated diagnostic and recovery logic. Interface volatility, infrastructure fluctuations, and data evolution create operational fragility requiring manual intervention. Self-healing architectures go beyond the matter of fixed automation logic and changing operational contexts. Smartness incorporated in the systems enables them to detect anomalies, figure out root causes, and execute the corrective actions without the help of a human. Finance processes, in particular, are made to be very efficient with this kind of system as they experience less downtime, and their workflow continuity improves. Payment processing workflows maintain transactional integrity through verification protocols, preventing duplicate charges. Reconciliation systems adapt to data source unavailability and schema evolution through intelligent caching and adaptive parsing. Reliability engineering practices establish comprehensive observability spanning traditional uptime metrics alongside recovery success rates and diagnostic accuracy measurements. Continuous learning frameworks ensure capabilities evolve through feedback loops capturing production outcomes. Machine learning models refine classification logic and expand remediation strategy repositories over time. Future advancements will enhance diagnostic intelligence through broader failure taxonomies and cross-process coordination mechanisms. Standardized reliability metrics will enable objective automation maturity comparisons across organizations. As financial automation scales in complexity, self-healing capabilities transition from competitive advantages to operational necessities. Sustained service quality and regulatory compliance increasingly depend on autonomous recovery mechanisms maintaining workflow continuity despite environmental disruptions.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Shashank Pasupuleti, "Robotic Process Automation for Enhancing Workflow Automation in Multi-System Environments," Journal of Advances in Developmental Research (IJAIDR), 2024. [Online]. Available: https://www.ijaidr.com/papers/2024/1/1134.pdf

[2] Jay Patel and Harshal Shah, "SOFTWARE ENGINEERING REVOLUTIONIZED BY MACHINE LEARNING-POWERED SELF-HEALING SYSTEMS," IRJEAS, 2021. [Online]. Available: https://www.irjeas.org/wp-content/uploads/admin/volume9/V9I1/IRJEAS04V9I1012103210000008.pdf

[3] WILLIAM VILLEGAS-CH et al., "Toward Intelligent Monitoring in IoT: AI Applications for Real-Time Analysis and Prediction," IEEE Access, 2024. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10471529

[4] Angelos Angelopoulos et al., "Tackling Faults in the Industry 4.0 Era—A Survey of Machine-Learning Solutions and Key Aspects," MDPI, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/1/109

[5] MICHEL NASS et al., "Similarity-based Web Element Localization for Robust Test Automation," ACM, 2023. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3571855

[6] Ashish Patil et al., "Exception Intelligence in High-Risk and High-Velocity Supply Chains: Typology, Playbooks, and Real-Time Resolution Systems," Journal of Advances in Developmental Research (IJAIDR), 2025. [Online]. Available: https://www.ijaidr.com/papers/2025/2/1475.pdf

[7] Flavin Cristian, "Understanding Fault-Tolerant Distributed Systems," Communications of the ACM, 1991. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/102792.102801

[8] Nilima James Rodrigues, "Transforming enterprise finance with data-centric architectures and platform integration," World Journal of Advanced

Engineering Technology and Sciences, 2025. [Online]. Available: https://www.researchgate.net/profile/Nilima-Rodrigues/publication/392764243

[9] Jason M. Pittman, "A MEASURE FOR LEVEL OF AUTONOMY BASED ON OBSERVABLE SYSTEM BEHAVIOR," arXiv, 2024. [Online]. Available: https://arxiv.org/pdf/2407.14975

[10] Jibinraj Antony et al., "Adapting to Changes: A Novel Framework for Continual Machine Learning in Industrial Applications," J Grid Computing, 2024. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s10723-024-09785-z.pdf