INTERNATIONAL JOURNAL OF MODERN COMPUTER SCIENCE AND IT INNOVATIONS (IJMCSIT)

eISSN: 3087-4297

Volume. 02, Issue. 11, pp. 01-11, November 2025"

Bridging The Gap: A Strategic Framework for Integrating Site Reliability Engineering with Legacy Retail Infrastructure

Dr. Felicia S. Lee
School of Computing, National University of Singapore, Singapore

Ivan A. Kuznetsov
Faculty of Computer Science, Higher School of Economics, Moscow, Russia

Article received: 05/09/2025, Article Revised: 06/10/2025, Article Published: 12/11/2025

ABSTRACT

Background: The retail sector faces intense pressure to ensure high availability and low latency, especially during peak traffic events. However, many established retailers operate on complex, monolithic legacy infrastructures that are inherently resistant to modern DevOps practices. Site Reliability Engineering (SRE), pioneered in cloud-native environments, offers a compelling model for managing reliability, yet its application in 'brownfield' legacy contexts is poorly understood.
Objectives: This study aims to (1) analyze the socio-technical friction points when implementing SRE principles within legacy retail organizations and (2) propose and evaluate a phased framework for this transition.
Methods: We employed a qualitative, multi-case study methodology, analyzing three anonymized retail organizations (grocery, e-commerce, department store) undergoing SRE adoption. Data was collected through 30 semi-structured interviews with engineering and leadership staff, supplemented by an analysis of internal documentation (postmortems, roadmaps, and monitoring data). We analyzed these cases through the lens of a proposed three-phase implementation framework: (1) Stabilize & Observe, (2) Automate & Abstract, and (3) Modernize & Scale.
Results: The findings indicate that the most significant barriers are cultural rather than technical, particularly the resistance to blameless postmortems and the adoption of error budgets. Defining meaningful Service Level Objectives (SLOs) for monolithic applications emerged as a complex initial hurdle. However, the study found that SRE-derived data (SLO breach reports, toil logs) provided a critical, objective language for prioritizing technical debt and de-risking modernization efforts, such as API abstraction and the introduction of new microservices.
Conclusion: SRE is a viable and necessary strategy for legacy retail, acting as a catalyst for incremental modernization. Successful adoption hinges on adapting SRE principles, prioritizing cultural change alongside technical automation, and using SRE metrics to bridge the divide between operations and development.

KEYWORDS

Site Reliability Engineering (SRE), Legacy Systems, Retail Technology, IT Modernization, DevOps, Service Level Objectives (SLOs), Toil Automation.

INTRODUCTION

1.1. The Reliability Imperative in Modern Retail

The contemporary retail landscape is defined by a relentless drive for digital transformation. The rise of e-commerce, mobile shopping applications, and omnichannel strategies has fundamentally altered customer expectations. Modern consumers expect seamless, instantaneous, and highly personalized experiences, regardless of whether they are interacting with a brand online, via a mobile app, or in a physical store. This expectation of flawless service translates into a stringent technical requirement: near-constant availability.

In this high-stakes environment, reliability is not merely an IT metric; it is a core business proposition. A minute of downtime during a promotional event or, most critically, a peak traffic period like Black Friday or a holiday sale, can result in millions in lost revenue, irreversible brand damage, and customer attrition. The latency of an inventory lookup, the success rate of a payment transaction, or the speed of a product page load are all direct drivers of conversion rates and customer loyalty. Consequently, the engineering challenge for retailers is no longer just to build features, but to operate services that are perpetually fast, available, and resilient.

1.2. The 'Legacy' Conundrum: When Infrastructure Resists Change

While consumer-facing applications present a veneer of modernity, the operational backbone of many established retailers is built upon legacy infrastructure. These 'brownfield' environments are often a complex tapestry of technologies accrued over decades. They are characterized by monolithic architectures, where core business functions—such as inventory management, order processing, and point-of-sale (POS) data synchronization—are tightly coupled within a single, massive codebase.

This architectural approach, common in systems developed in the 1980s and 1990s, presents profound challenges. These systems often run on bare-metal servers or early-generation virtual machines, managed with manual processes and infrequent, high-risk "big bang" release cycles. Observability is typically poor, limited to basic server health metrics (e.g., CPU, memory) rather than the user-centric performance indicators required by modern services. Data is often siloed in proprietary mainframe or AS/400 databases, accessible only through arcane protocols. This technical debt makes the infrastructure brittle, difficult to scale, and terrifyingly risky to modify. The very systems responsible for the most critical business functions are the least adaptable to the demands of the modern reliability imperative.

1.3. The Emergence of Site Reliability Engineering (SRE)

In response to the challenge of managing massive, distributed systems at scale, Google pioneered the discipline of Site Reliability Engineering (SRE). SRE is a prescriptive approach that treats operations as a software engineering problem. As described by its progenitors, SRE is "what happens when you ask a software engineer to design an operations team." It diverges sharply from traditional IT operations models (like ITIL), which tend to silo development and operations teams and manage change through rigid, ticket-based processes.

The SRE model is built on several core tenets. First is the primacy of Service Level Objectives (SLOs), which are specific, measurable, and user-centric targets for service performance (e.g., "99.9% of checkout requests will be successful"). These SLOs are not aspirations; they are hard data points used to guide engineering decisions. Second is the error budget, the mathematical inverse of the SLO ($1 - SLO$). This budget represents the acceptable level of failure for a service. If a service exceeds its error budget, all new feature development is frozen, and all engineering resources are redirected to improving reliability. This mechanism creates a self-regulating, data-driven balance between innovation and stability.

Other key SRE practices include the aggressive automation of toil (manual, repetitive, automatable operational work) and the adoption of a blameless postmortem culture. This culture shifts the focus of incident investigation away from "human error" and toward understanding the systemic and programmatic failures that allowed an incident to occur, fostering a climate of continuous learning and improvement

1.4. Literature Gap and Research Problem

The principles of SRE, along with the broader DevOps movement, have been transformative for organizations born in the cloud. An abundance of literature describes how to implement SRE in 'greenfield' projects characterized by microservice architectures, containerized workloads managed by platforms like Kubernetes, and robust public cloud infrastructure. These environments are, by their nature, designed for the kind of abstraction, automation, and rapid iteration that SRE thrives upon.

However, a significant gap exists in the academic and professional literature regarding the application of SRE to the 'brownfield' world of legacy systems. Recent industry analyses highlight that legacy modernization remains a top priority for CIOs, yet the operational risks involved are a primary barrier to progress. The central problem is a fundamental mismatch: How can organizations apply principles forged in the world of ephemeral, scalable, cloud-native services to the rigid, monolithic, on-premise infrastructure that still powers the global retail economy?

Applying "by the book" SRE in such an environment is often untenable. How does one define a granular SLO for a function buried deep within a 20-year-old COBOL mainframe application? How is an error budget enforced when the release cycle is six months long? How can automation be implemented when the systems lack modern APIs? There is a critical need for a structured, adaptive framework that bridges the gap between SRE theory and legacy reality.

1.5. Research Objectives and Article Structure

This study seeks to address this gap by investigating the implementation of SRE within legacy-heavy retail organizations. The research is guided by three primary objectives:

1. To identify and analyze the specific socio-technical friction points that arise when SRE principles (SLOs, error budgets, blameless postmortems, toil automation) are introduced into traditional, operations-focused IT environments in the retail sector.

2. To propose a phased methodological framework designed specifically for the incremental adoption of SRE in organizations managing legacy infrastructure.

3. To evaluate the application of this framework through a qualitative, multi-case analysis of SRE adoption efforts, identifying common challenges and success factors.

To achieve these objectives, this article follows the IMRaD structure. The Methodology section details our qualitative, multi-case study design, the criteria for case selection, and the data collection and analysis procedures. It also provides a comprehensive, detailed breakdown of the proposed three-phase implementation framework that serves as the analytical lens for the study. The Results section presents the empirical findings from our three case studies, structured according to this phased framework. The Discussion section synthesizes these findings, interprets their implications for theory and practice, and validates the proposed framework. Finally, the Conclusion summarizes the study's contributions, acknowledges its limitations, and suggests avenues for future research.

2. METHODOLOGY

2.1. Research Design: A Qualitative, Multi-Case Approach

To investigate the complex, nuanced, and context-dependent phenomena of organizational and technical change, this study employed a qualitative, multi-case study research design. SRE adoption is not merely a technical tooling problem; it is a profound socio-technical transformation that involves shifts in culture, team structure, political power, and engineering philosophy. A quantitative approach, while useful for measuring specific outcomes (like SLO adherence), would fail to capture the process of adoption and the nature of the barriers encountered.

A comparative case study approach allows for deep, contextualized exploration of real-world implementations. By comparing the experiences of different organizations, we can identify emergent patterns, common challenges, and divergent strategies, leading to a richer, more transferable (though not generalizable) set of findings. This approach is ideally suited for answering "how" and "why" questions, such as how organizations adapt SRE principles and why certain adaptations succeed or fail in a legacy context.

2.2. Case Selection Criteria

We purposefully selected three large, anonymized retail organizations for this study, hereafter referred to as "Case A," "Case B," and "Case C." Selection was based on a specific set of criteria designed to ensure relevance to the research questions:

1. Sector: The organization must derive the majority of its revenue from retail operations (e.g., grocery, fashion, department store).

2. Legacy Infrastructure: The organization's core business functions (e.g., inventory, POS, or core e-commerce) must rely on systems identified by its own IT leadership as "legacy" (e.g., monolithic architecture, mainframe, pre-cloud COTS platforms).

3. SRE Adoption Status: The organization must have formally initiated an SRE adoption program within the last 24-36 months, moving beyond theoretical discussion into active implementation.

4. Access: The organization must grant access for researchers to conduct semi-structured interviews and review non-sensitive internal documentation.

The three selected cases provided a diverse cross-section of the retail legacy challenge:

- Case A (GroceryCo): A large, national grocery chain. Its primary legacy challenge is a mainframe-based inventory and supply chain system. This system is highly reliable but opaque, and all new development (e.g., online ordering) is severely bottlenecked by the mainframe's limited, batch-oriented integration points.

- Case B (FashionRetail): A "clicks-and-mortar" fashion retailer. Its legacy challenge is a monolithic, first-generation e-commerce platform (circa 2005) that has been heavily customized. It suffers from severe performance and scaling issues during promotional events.

- Case C (StoreCo): A national department store chain. Its legacy challenge is a highly disparate and aging set of in-store Point-of-Sale (POS) systems across hundreds of stores. These systems frequently suffer from data synchronization failures with the central order management system.

2.3. Data Collection

Data collection occurred over a 12-month period and employed a triangulated approach to ensure richness and validity.

- Semi-Structured Interviews (N=30): We conducted 30 interviews, with 10 participants from each case organization. Participants were selected to provide a holistic view, including:
  - Traditional Operations/Infrastructure Staff (n=12)
  - Software Developers (n=9)
  - Newly hired or designated SREs (n=5)
  - IT Leadership (CIO, VP of Infrastructure) (n=4)

Interviews lasted 60-90 minutes and were guided by a protocol focused on the SRE adoption process, perceived challenges, cultural shifts, and the specific application of SRE principles (SLOs, postmortems, etc.).

- Document Analysis: We were granted access to a repository of internal documents, which provided concrete artifacts of the SRE transformation. These included:
  - Blameless postmortem reports (n=15, anonymized)
  - SRE team charters and mission statements.
  - Internal wiki pages and Confluence articles defining SLOs and error budget policies.
  - Excerpts from monitoring dashboards (screenshots) and internal project roadmaps.
- Participant Observation (Limited): Where permitted, researchers attended (as non-participant observers) SRE team planning meetings and incident review (postmortem) meetings (n=6). This provided direct insight into the team dynamics and cultural practices being enacted.

2.4. A Proposed Phased Implementation Framework (The Analytical Model)

To guide the data analysis and structure the investigation, we first synthesized a methodological framework from preliminary literature and exploratory interviews. This "Phased Implementation Framework" posits that SRE cannot be adopted in a "big bang" in legacy environments. Instead, it requires a gradual, incremental approach. This framework, detailed below, serves as the core "method" proposed by this study, which the Results section will then use as an analytical lens to evaluate the three case studies.

The framework consists of three sequential, overlapping phases:

2.4.1. Phase 1: 'Stabilize & Observe' – Establishing Foundational Reliability

This initial phase is the most critical and focuses on shifting the organization from a reactive, "firefighting" posture to a proactive, data-driven one. It does not involve significant automation or modernization but lays the cultural and observational groundwork for all subsequent efforts.

- 2.4.1.1. Defining Critical User Journeys (CUJs) in a Monolithic Context: Before metrics can be established, the team must define what to measure. In a monolithic system, this is exceptionally difficult. A CUJ is a user-centric path through the system that delivers value (e.g., "User searches for product," "User adds item to cart," "User completes checkout"). In a legacy context, this requires identifying these journeys as they traverse the monolith. For example, a POS "checkout" journey might involve the local terminal, a store-level server, and a central database, all within one application stack. The SRE team must first map these critical flows, often through reverse-engineering or tribal knowledge.
- 2.4.1.2. Selecting Service Level Indicators (SLIs): An SLI is the raw metric used to measure a CUJ (e.g., latency, error rate). This framework argues against using traditional legacy metrics like server CPU or memory utilization as SLIs, as these are symptoms, not user-facing indicators. The focus must be on symptom-based monitoring. The most effective SLIs for legacy systems are typically "black-box" metrics measured at the perimeter:
  - Availability: The percentage of valid requests that receive a non-error response, measured at the load balancer or application entry point.
  - Latency: The time taken to service a request, also measured at the perimeter.
  - Success Rate: A more nuanced availability metric, such as the percentage of "checkout" API calls that return "success" rather than "payment failed" or "inventory error."
- 2.4.1.3. Negotiating Initial Service Level Objectives (SLOs): An SLO is the target for an SLI (e.g., "99.9% of checkout requests will be successful over a 28-day window"). This framework posits that setting aspirational SLOs (e.g., "five nines" or 99.999%) in a legacy environment is counter-productive. It creates immediate failure, demoralizes the team, and exhausts the error budget instantly, rendering the concept useless. Instead, the initial SLOs should be data-driven and achievable. Teams should first measure their baseline

performance for 30-60 days (e.g., "We are currently 99.5% successful") and set the initial SLO just above that baseline (e.g., 99.7%). The SLO can then be progressively tightened as reliability improves. This negotiation is a political act, aligning Development, Operations, and the Business on an acceptable level of reliability.

- 2.4.1.4. Instituting the Blameless Postmortem Culture: This is the most significant cultural component of Phase 1. Traditional ITIL-based organizations practice "Root Cause Analysis" (RCA), which often concludes with "human error" and seeks an individual to blame. SRE replaces this with a blameless postmortem culture. This framework defines a structured process for this:

1. An incident is declared and resolved.

2. Within 72 hours, a postmortem meeting is held, moderated by a neutral party.

3. The meeting is forbidden from using language of blame ("Who...?", "Why did you...").

4. The focus is on a factual timeline: What happened? What was the impact? How was it detected? How was it remediated?

5. The analysis focuses on systemic causes: Why was the system brittle? Why did monitoring fail to detect it? Why was the remediation process slow?

6. The output is a document with "Corrective Actions"—specific, assigned, and time-boxed engineering tasks to prevent the class of problem from recurring. This document is public to the entire engineering organization.

2.4.2. Phase 2: 'Automate & Abstract' – Reclaiming Engineering Time

With foundational observability and a learning culture in place, Phase 2 focuses on using that data to actively reduce operational load and de-risk the legacy system.

- 2.4.2.1. The 'Toil Budget' and Identification: Toil is defined as operational work that is manual, repetitive, automatable, tactical (not strategic), and scales linearly with service growth. It is the "firefighting" and "grunt work" that consumes traditional Ops teams. In this phase, SREs are mandated to track their time, specifically logging hours spent on toil. This data is used to establish a "toil budget" (e.g., an SRE should spend no more than 50% of their time on toil). Any toil exceeding this budget must be automated. This creates a data-driven incentive to build software that automates operations. Examples in retail include: manually restarting failed batch inventory jobs, clearing caches, or provisioning user accounts.

- 2.4.2.2. The Error Budget Policy: As defined in Phase 1, the SLO creates an error budget. In Phase 2, this budget is enforced as a policy. This is the SRE's primary mechanism for balancing reliability and new features. The policy, agreed upon by leadership, states: If a service has consumed its 28-day error budget (due to outages or performance degradation), all new feature releases for that service are frozen. All development resources for that service are redirected to work on the "Corrective Actions" backlog from the postmortems until the service is stable and operating within its SLO. This is a radical political tool in legacy environments, as it uses objective data to halt the business's demand for "more features" in favor of the stability the business also demands.

- 2.4.2.3. The 'Strangler' API Abstraction Layer: This is the primary technical strategy of Phase 2. Modifying the core of a legacy monolith is unacceptably risky. The "Strangler Fig" pattern (or API Abstraction) involves "wrapping" the legacy system in a modern, well-documented, reliable API layer. For example, instead of allowing a new mobile app to query the mainframe inventory database directly, it queries a modern REST API. This API, in turn, handles the complex and brittle interaction with the mainframe. This strategy achieves two goals: (1) It abstracts the legacy system, making it easier and safer for new applications to consume its data. (2) It creates a "seam" where, in the future, the legacy system can be replaced piece by piece behind the API without the front-end services ever knowing.

2.4.3. Phase 3: 'Modernize & Scale' – Data-Driven Evolution

In this phase, the SRE team transitions from stabilizing the past to actively building the future, using the data from Phases 1 and 2 to guide the modernization strategy.

- 2.4.3.1. Using SRE Data to Prioritize Modernization: Legacy modernization is expensive and complex. The most common question is: "Where do we start?" The SRE data provides the answer. The backlog of postmortem corrective actions, combined with SLO breach reports and toil logs, creates a data-driven, prioritized roadmap for modernization. If 60% of SLO breaches are related to the monolithic payment processing module, that module becomes the top priority for being "strangled" (per 2.4.2.3) and rewritten as a modern microservice. This moves the modernization debate from one based on opinion and "gut feel" to one based on objective data about user pain.

- 2.4.3.2. Introducing Cloud-Native Tooling for New Services: This framework explicitly argues against attempting to containerize the legacy monolith. It is not a "lift and shift" model. Instead, SREs manage a hybrid environment. The legacy stack is left in place, managed by the new API abstractions and automation scripts. All new services (e.g., the new payment microservice) are built on a modern, containerized platform (like

Kubernetes or a cloud provider's equivalent). The SRE team's role expands to manage the reliability of this new platform and, critically, the seams and dependencies between the new microservices and the legacy API-wrapped monoliths.

- 2.4.3.3. Integrating AIOps for Legacy Observability: Many legacy systems produce unstructured or non-standard logs (e.g., mainframe job logs, COBOL error codes) that modern observability tools cannot easily parse. Phase 3 introduces the use of AIOps (AI for IT Operations) as an advanced solution. Machine learning models can be trained on these unstructured logs to identify anomalous patterns before they trigger a user-facing SLO breach. This provides predictive failure analysis for the systems that are the least observable, completing the observability journey that began in Phase 1.

2.5. Data Analysis

We employed a thematic analysis approach to analyze the interview transcripts, observation notes, and internal documents. The initial coding structure was based on the three phases of our proposed framework (Stabilize & Observe, Automate & Abstract, Modernize & Scale) and the core SRE tenets (SLOs, Error Budgets, Postmortems, Toil).

Two researchers independently coded a subset of the data to establish an initial codebook and ensure inter-coder reliability. The full dataset was then coded, with emergent themes (e.g., "Cultural Resistance," "Management Buy-in," "Tooling Mismatch") being added to the codebook. A cross-case synthesis was then performed, comparing how each of the three organizations (GroceryCo, FashionRetail, StoreCo) navigated the challenges within each phase of the framework. This comparative analysis allowed us to refine the framework and identify the most critical success factors and failure modes.

3. RESULTS

The analysis of the three case studies revealed a consistent, albeit challenging, journey of SRE adoption that broadly aligned with the proposed three-phase framework. However, the progression was not linear, and each organization encountered significant friction, particularly in the cultural and political realms.

3.1. Overview of SRE Adoption Trajectories

- Case A (GroceryCo): Began SRE adoption driven by a new CIO. The primary goal was to de-risk the mainframe-based supply chain to support a new online ordering initiative. They focused heavily on Phase 1 (SLOs, Postmortems) and Phase 3 (AIOps for mainframe logs) but struggled with Phase 2 (Automation), as the pool of engineers with both mainframe and modern automation skills was small.

- Case B (FashionRetail): Adoption was a grassroots effort from the engineering team, frustrated by constant "firefighting" during sales. They excelled at Phase 2 (Toil Automation, Error Budgets), using the error budget as a "weapon" to force leadership to pause features and address technical debt. They struggled with Phase 1, finding it hard to get business buy-in for blameless postmortems.

- Case C (StoreCo): Adoption was top-down, part of a major digital transformation. Their primary success was in Phase 2 (API Abstraction), successfully "wrapping" their disparate POS systems in a unified API. This immediately stabilized data synchronization. They faced their greatest challenge in Phase 1 (SLOs), as the performance of the in-store systems was highly variable and difficult to measure centrally.

3.2. Phase 1 Analysis: 'Stabilize & Observe'

This phase was universally identified as the most difficult and most important.

3.2.1. The Challenge of Defining SLOs for Monoliths

All three cases struggled to move from server-based metrics to user-centric SLOs.

- Difficulty in Measurement: In Case C (StoreCo), the team initially tried to set an SLO for "POS Transaction Time." They discovered that the transaction time was recorded in a local log on 1,200 different store servers, in different formats, and only uploaded nightly. It was impossible to measure in real-time. They were forced to redefine their SLI to "API success rate" for the central order management system receiving the data, which was a "good enough" proxy for reliability, though not a true measure of the user's (cashier's) experience.

- Negotiating Baselines: In Case B (FashionRetail), the e-commerce platform was notoriously slow. The engineering team measured the 90th percentile (p90) latency for "Add to Cart" at 3.5 seconds. The business wanted an SLO of 800ms. The SRE team, following the framework, argued that this was aspirational and would fail immediately. As one SRE lead stated, "We had to show them the data and say, 'The system is this slow. We can't promise 800ms tomorrow. Let's set the SLO at 3.0 seconds, and when we are consistently meeting that, we can tighten it to 2.5 seconds.' It was the first honest conversation about performance we'd ever had."

- Proxy Metrics for Mainframes: In Case A (GroceryCo), measuring the "latency" of a batch inventory update was nonsensical. The team shifted to a

different CUJ: "Inventory Data Freshness." They defined an SLI as the time elapsed between a sale occurring in-store and that inventory change being reflected in the e-commerce system. Their initial SLO was "99% of inventory updates will be reflected online within 15 minutes." This was a meaningful, user-centric SLO for a legacy batch-oriented system.

3.2.2. Cultural Barriers to Blameless Postmortems

This was, by far, the most significant socio-technical hurdle.

- Resistance from ITIL Culture: Case A (GroceryCo) had a deeply embedded ITIL culture. The existing "Problem Management" team saw SRE's blameless postmortems as a "lack of accountability." An Operations Manager noted, "For 20 years, my job has been to find out who made the change that broke the system. You are asking me to ignore that and talk about 'systemic weakness.' It feels like we are letting people off the hook." It took direct, sustained executive sponsorship from the CIO to enforce the blameless model and protect the engineers who were participating honestly.

- Fear of Admitting Failure: In all cases, engineers were initially silent in postmortem meetings, fearing retribution. The "blameless" declaration was met with cynicism. In Case B (FashionRetail), the SRE lead had to "sacrifice" themselves first. They moderated a postmortem about an incident they had caused, publicly analyzing the systemic issues (e.g., "The script I ran was dangerous, and the system should have prevented it from running in production. The monitoring should have caught the impact faster.") This act of vulnerability was cited as a turning point in building psychological safety.

- Shifting from 'Fix' to 'Prevent': The output of traditional RCAs was almost always "re-train staff" or "add checklist item." The output of the new SRE postmortems was engineering work. In Case C (StoreCo), a data sync failure postmortem produced a corrective action: "Build a reconciliation tool that idempotently retries failed POS uploads." This was a fundamental shift from blaming an operator for a manual data-entry error to building a system that was resilient to that class of error.

3.3. Phase 2 Analysis: 'Automate & Abstract'

Once baseline stability and a learning culture began to form, teams found significant value in Phase 2.

3.3.1. Successes in Toil Reduction

Identifying and automating toil was a major "quick win" that built momentum for SRE.

- Quantifying the Pain: In Case B (FashionRetail), the SRE team was required to log toil. They discovered that two engineers were spending a combined 30 hours per week manually clearing a specific product cache on the monolithic e-commerce platform. As one developer said, "We all knew it was a pain, but when leadership saw '30 hours' on a spreadsheet, they finally approved the two weeks of engineering time to build an automated cache invalidation service."

- Empowering Operations: In Case A (GroceryCo), the mainframe operations team had a 50-page "runbook" for manually restarting failed nightly batch jobs. The new SRE team (a mix of ops and developers) spent three months writing Python scripts to automate 80% of that runbook. This freed the mainframe operators to learn new skills and contribute to observability projects, rather than just "babysitting the batch."

3.3.2. The Error Budget as a Political Tool

This was the most controversial, yet most powerful, SRE concept.

- Forcing the Reliability Conversation: In Case B (FashionRetail), the e-commerce platform chronically breached its newly established latency SLO (p90 > 3.0s), exhausting its error budget two weeks into every month. The marketing team wanted to launch a new personalization engine. Following the new policy, the SRE team announced a "feature freeze." The VP of Marketing escalated to the CTO, who pointed to the data: "We cannot add new features to a system that is already failing our users." The personalization project was paused for one quarter, and engineering resources were redirected to optimize the database query cache and image rendering pipeline. The p90 latency dropped to 2.2 seconds. This event, cited by all participants in Case B, cemented the SRE team's authority and the value of the error budget.

- Failure to Enforce: In Case C (StoreCo), leadership was unwilling to enforce the error budget policy. The POS API service was consistently breaching its SLOs, but the business "accepted the risk" and continued to push for new in-store features. The SRE team's alerts became "noise," and morale plummeted. An SRE there noted, "An error budget without teeth is just another dashboard. It has no power. We measure the failures, but we aren't allowed to stop them from happening."

3.3.3. API Abstraction as a Decoupling Strategy

The "Strangler Fig" pattern was a universally successful technical strategy.

- De-Risking the POS: Case C (StoreCo) provides the clearest example. The project to replace all 1,200 POS terminals was a non-starter (estimated $50M+ cost).

Instead, the SRE team led the development of a modern, central "POS Abstraction API." This API accepted data from all the different, aging POS models and formats, transformed it, and fed it reliably to the central systems. This $1M project solved the data synchronization failures within six months. It also meant that new services, like a mobile "scan-and-go" app, could be built to talk to this single, modern API, completely ignorant of the legacy mess behind it.

- Unlocking Mainframe Data: In Case A (GroceryCo), the new online ordering system needed real-time inventory. The mainframe team (now part of the SRE org) built a "change-data-capture" service that streamed inventory changes from the mainframe database to a Kafka queue, which then populated a modern API. This "Inventory API" provided sub-second inventory data to the e-commerce site, effectively turning a 50-year-old batch system into a (near) real-time service without touching the core COBOL code.

3.4. Phase 3 Analysis: 'Modernize & Scale'

This phase was the least mature in all cases, but the trajectory was clear.

3.4.1. Prioritizing Modernization via SRE Data

SRE data became the new language of prioritization.

- From "Gut Feel" to Data-Driven: Before SRE, the modernization roadmap in Case B (FashionRetail) was, as one director put it, "a shouting match between the marketing team and the highest-paid engineer." After SRE, the postmortem corrective action backlog and the SLO breach reports provided an objective list. The team analyzed six months of data and found that 55% of all error budget consumption was linked to the monolithic "Checkout" service. This service was immediately targeted as the first piece of the monolith to be "strangled" and rewritten as a separate microservice.

- Justifying the Cost: In Case A (GroceryCo), the SRE team used toil data to justify modernization. They calculated the engineering hours spent manually managing the batch jobs (per 3.3.1) and presented the annual, fully-loaded cost to the CIO. This cost was so high that it easily justified the budget for a modern workload automation scheduler, demonstrating a clear ROI for modernization.

3.4.2. The Hybrid Infrastructure Challenge

All organizations were now managing a hybrid environment, which created new SRE challenges.

- Managing the Seams: As Case B (FashionRetail) began building its new "Checkout" microservice on Kubernetes, the SRE team's focus shifted. They were now responsible for both the legacy monolith (on VMs) and the new Kubernetes platform. The most complex incidents were those that occurred at the "seam" between the two—for example, when the new microservice called an old API on the monolith, causing a database connection pool to exhaust. The SRE team had to have "full-stack" knowledge of both worlds, a skillset that was extremely difficult to hire or train.

- Tooling Gaps: The teams' new, modern observability tools worked perfectly for the Kubernetes side but struggled to ingest and correlate data from the legacy side. This created a "split-brain" monitoring scenario, making it hard to trace a single user request (CUJ) as it hopped from a new microservice to a legacy API and back.

3.4.3. Early Adoption of AIOps

Case A (GroceryCo) was the only organization actively experimenting with AIOps, with promising results.

- Predictive Failure for Mainframes: The mainframe system produced tens of thousands of cryptic log messages nightly. It was impossible for a human to review. The SRE team partnered with a vendor to train an AIOps model on these logs. After three months, the model began to successfully identify anomalous log patterns 1-2 hours before the associated batch job would fail. This allowed the SRE team to proactively intervene and prevent a failure that would have previously breached the "Inventory Data Freshness" SLO. An operations engineer remarked, "It's the first time in my career we've ever fixed a mainframe problem before it happened."

4. DISCUSSION

The findings from these three case studies provide significant insights into the adaptation of Site Reliability Engineering within legacy retail environments. This discussion synthesizes these findings to address the research objectives, validate the proposed framework, and explore the broader implications for both theory and practice.

4.1. SRE in Legacy Contexts: An Interpretive Synthesis

Our first research objective was to identify the socio-technical friction points. The results overwhelmingly indicate that SRE adoption in legacy organizations is primarily a cultural and political challenge, not a technical one. The technical hurdles, while significant (e.g., measuring monoliths, automating mainframes), were solvable with dedicated engineering effort. The socio-technical barriers, however, proved far more resilient.

The resistance to blameless postmortems, as seen in Case A, highlights the deep-seated nature of a "blame culture"

within traditional ITIL-based operations. SRE demands psychological safety, a willingness to expose systemic weakness without fear of reprisal. This is a radical departure from an operations model built on "change control" and "root cause analysis" that often seeks to identify a human error. Without top-down executive sponsorship to protect engineers and enforce the blameless model, the entire SRE cultural transformation fails.

Similarly, the error budget's power, as seen in Case B, is entirely political. It is a policy tool that reifies reliability as a first-class, non-negotiable feature, equal to (or greater than) new business features. Where leadership was willing to enforce the "feature freeze" (Case B), SRE succeeded in creating a self-regulating system that balanced innovation and stability. Where leadership wavered (Case C), the error budget became "performative metrics," and SRE failed to gain traction, devolving into just another monitoring team.

This confirms that SRE is not a set of tools that can be purchased. It is an organizational philosophy that must be adopted, and this adoption directly challenges existing power structures and deeply ingrained cultural norms.

The integration of Site Reliability Engineering principles within legacy retail ecosystems demands a proactive approach to fault tolerance and observability. Kumar Tiwari et al. (2025) emphasized the value of Chaos Engineering in strengthening resilience and ensuring consistent service reliability in distributed environments. This approach provides a practical foundation for bridging traditional system limitations with modern reliability engineering practices, enabling seamless scalability, predictive monitoring, and operational adaptability across hybrid infrastructures.

4.2. Validating the Phased Implementation Framework

The second research objective was to propose and evaluate a phased framework. The experiences of the three cases largely validate the proposed three-phase model (Stabilize & Observe; Automate & Abstract; Modernize & Scale) as a viable pathway.

The findings clearly show that Phase 1: 'Stabilize & Observe' is the non-negotiable foundation. Organizations that attempt to skip straight to automation (Phase 2) or modernization (Phase 3) are likely to fail. As one participant noted, "If you automate a broken process, you just get a faster broken process." Without first establishing what is "normal" (SLOs) and a process for learning from failure (postmortems), any subsequent automation or modernization efforts are based on guesswork. The struggles of all three cases with defining SLOs and implementing postmortems underscore that this initial, non-technical phase requires the most focus and executive support.

Phase 2: 'Automate & Abstract' was shown to be the "value-delivery" phase. Toil automation (Case B) provided immediate ROI in reclaimed engineering hours, building credibility for the SRE team. The API abstraction strategy (Case C) was the key technical unlock, allowing organizations to "neuter" the risk of their legacy systems without the astronomical cost and risk of a full rewrite. This "Strangler Fig" approach is perhaps the single most critical technical pattern for SRE in a legacy context.

Phase 3: 'Modernize & Scale' was correctly identified as the long-term, data-driven outcome. The findings confirm that SRE is not an alternative to modernization; it is the catalyst for it. The data generated by SRE practices (SLO breaches, toil logs, postmortem actions) provides the objective business case required to secure funding and prioritize the modernization roadmap, moving it from opinion-based to data-driven. The emergence of hybrid infrastructure management and AIOps (Case A) confirms this as the "future state" of a successful legacy SRE team.

One key refinement to the model, based on the findings, is the need to explicitly add a parallel "Executive & Cultural" track. The framework as proposed is largely technical and procedural. The results suggest a parallel track (e.g., "Secure Executive Sponsorship," "Train Middle Management," "Evangelize Psychological Safety") is required to address the socio-technical barriers identified.

4.3. SRE as a 'Brownfield' Modernization Catalyst

This study contributes to the broader literature on legacy modernization. Current industry reports emphasize the urgency of modernization, but often focus on "lift-and-shift" to cloud or full rewrites, both of which are high-risk. Our findings propose SRE as a third, more incremental and data-driven path.

By "wrapping" the legacy system in SLOs and an API abstraction layer, SRE effectively stabilizes the "brownfield" environment. It makes the unreliable predictable. The error budget then provides a non-confrontational, objective language for developers and operations—who are often organizationally siloed—to negotiate the "cost" of new features versus the "cost" of technical debt. SRE, in this light, is the operational discipline that de-risks the long, multi-year journey of incremental modernization, allowing retailers to keep servicing customers on their old infrastructure while safely building the new.

The adoption of SRE principles, even in non-cloud-native environments, aligns with the core philosophies of modern software development and DevOps. It forces organizations to treat infrastructure as code, automate relentlessly, and use data to make decisions, effectively

bringing legacy systems into the modern engineering fold.

4.4. Limitations of the Study

This study's findings should be interpreted in light of several limitations.

1. Qualitative Nature: As a qualitative, multi-case study, the findings are context-rich but not statistically generalizable. The experiences of these three retailers may not be representative of all retail organizations or other industries (e.g., finance, healthcare) with different legacy challenges.

2. Anonymization: To gain access, we had to heavily anonymize the cases. This prevents a detailed discussion of specific tooling choices (e.g., which APM vendor, which automation platforms) or the specific nature of their mainframe or POS technologies, which could be relevant factors.

3. Time Horizon: The study captured the first 24-36 months of SRE adoption. SRE is a multi-year, or even perpetual, journey. The long-term outcomes of these transformations, particularly the success of Phase 3, remain to be seen.

4. Selection Bias: The organizations that agreed to participate are, by definition, those that are open to reflection and likely more progressive. We did not capture the experiences of organizations that attempted SRE and failed completely, which could provide important counter-narratives.

4.5. Conclusion and Future Research

This study addresses a critical gap in the literature by examining the application of Site Reliability Engineering principles within the challenging context of legacy retail infrastructure. Our findings demonstrate that while "by the book" SRE is a poor fit for monolithic, on-premise systems, an adapted, phased approach can be highly successful.

We found that the primary barriers to adoption are socio-technical, centered on the cultural shift to blamelessness and the political enforcement of error budgets. We proposed and validated a three-phase framework (1) Stabilize & Observe, (2) Automate & Abstract, (3) Modernize & Scale—as a viable roadmap. The key technical strategies identified are the "wrapping" of monoliths with modern SLOs and API abstraction layers, while the key cultural strategy is the data-driven negotiation enabled by error budgets. This study concludes that SRE is not only possible in legacy retail but is a necessary operational discipline that acts as a powerful, data-driven catalyst for incremental modernization.

Future research should build on these qualitative findings. Longitudinal studies are needed to track SRE maturity and its impact on business metrics (e.g., revenue, customer satisfaction) over a 5- to 10-year horizon. Quantitative studies could compare the adherence to SLOs and the frequency of incidents in legacy organizations that adopt SRE versus those that follow traditional ITIL models. Finally, more research is needed on the emerging challenge of managing the hybrid SRE environment, developing the tools and team structures needed to ensure reliability across the widening seam between the legacy and cloud-native worlds.

REFERENCES

1. Allspaw, J. (2017). Blameless PostMortems and a Just Culture: A Guide to Incident Investigation. Etsy Engineering. https://codeascraft.com
2. Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media.
3. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. Communications of the ACM, 59(5), 50–57. https://doi.org/10.1145/2890784
4. Gartner. (2023). Predicts 2023: Legacy Systems Modernization Strategies for CIOs. Gartner Research.
5. Kumar Tiwari, S., Sooraj Ramachandran, Paras Patel, & Vamshi Krishna Jakkula. (2025). The Role of Chaos Engineering in Enhancing System Resilience and Reliability in Modern Distributed Architectures. International Journal of Computational and Experimental Science and Engineering, 11(3). https://doi.org/10.22399/ijcesen.3885
6. Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations. IT Revolution Press.
7. Krief, M. (2019). Learning DevOps: Continuously Deliver Better Software. Packt Publishing.
8. OpenSLO. (2021). Open Specification for SLOs. https://openslo.com
9. Thongmak, M. (2022). Applying AI in IT Operations: Anomaly Detection and Incident Prediction in Legacy Systems. Journal of Information Technology Management, 33(1), 35–42.
10. Woodcock, S. (2020). Automating Legacy Systems: Practices and Pitfalls. IEEE Software, 37(4), 67–73.

https://doi.org/10.1109/MS.2020.2996582

11. Zero-Trust Architecture in Java Microservices. (2025). International Journal of Networks and Security, 5(01), 202-214. https://doi.org/10.55640/ijns-05-01-12

12. Vikram Singh, 2025, Policy Optimization for Anti-Money Laundering (AML) Compliance using AI Techniques: A Machine Learning Approach to Enhance Banking Regulatory Compliance, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 14, Issue 04 (April 2025)