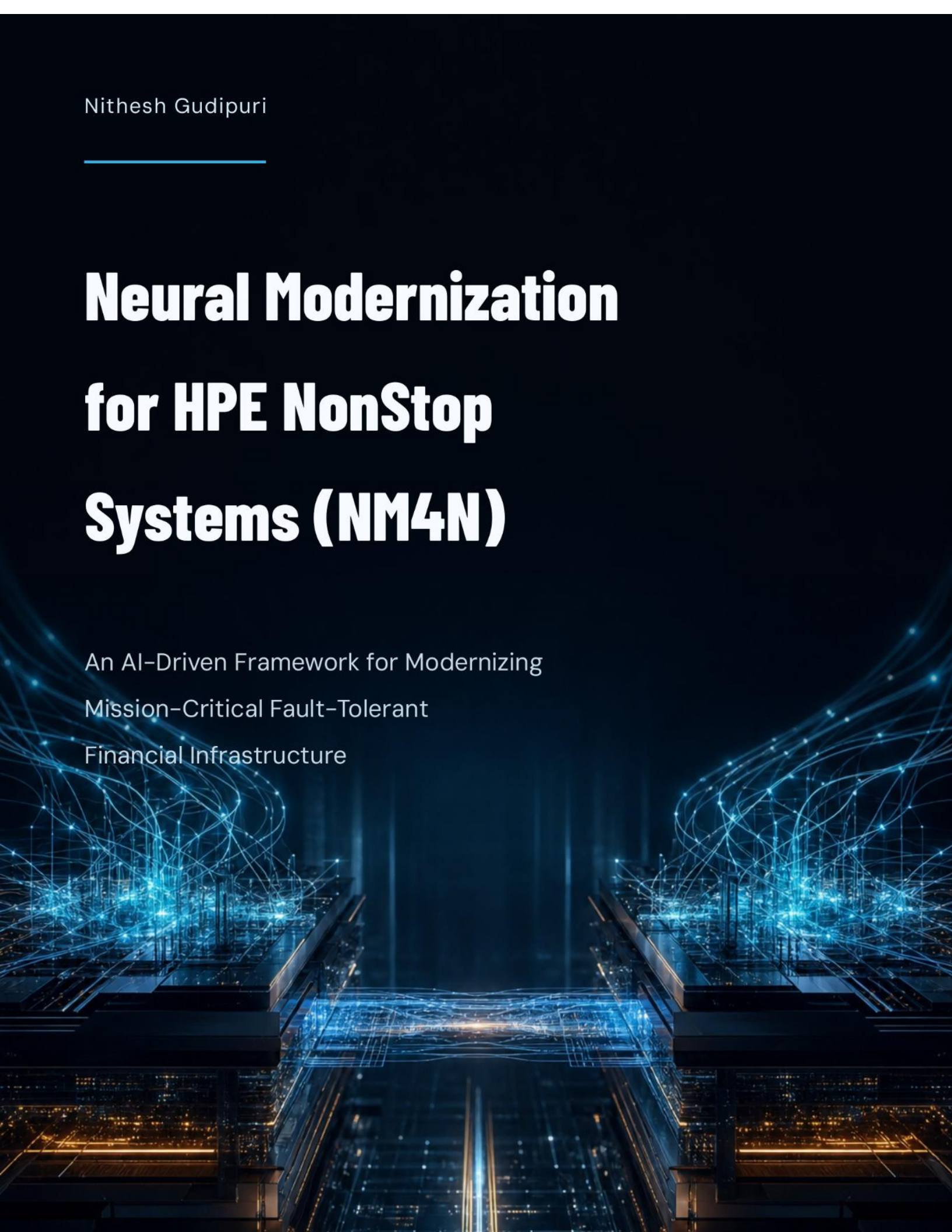
Nithesh Gudipuri

Neural Modernization for HPE NonStop Systems (NM4N)

An AI-Driven Framework for Modernizing Mission-Critical Fault-Tolerant Financial Infrastructure



Publication Partner:
International Journal of Scientific and Research Publications (ISSN: 2250-3153)

Neural Modernization for HPE NonStop Systems (NM4N)

An AI-Driven Framework for Modernizing Mission-Critical Fault-Tolerant Financial Infrastructure

Nithesh Gudipuri

Publishing Partner:
IJSRP Inc.
www.ijsrp.org

ISSN 2250-3153
9 772250 315302

Preface

The HPE NonStop platform has been at the center of global financial-services and telecommunications infrastructure for four decades. The platform's combination of process pairs, hardware-anchored transaction commit ordering, message-based inter-process communication, and the Transaction Management Facility deliver an availability level that no general-purpose runtime has been able to match. Six of the ten largest global retail banks, more than half of the Global 500 banks, three quarters of the Global 500 telecommunications operators, and the central securities depositories whose annual settlement turnover exceeds €800 trillion, run their mission-critical workloads on the platform.

The same combination that made the platform indispensable has made its modernization intractable for two decades. The TAL-competent engineering cohort has compressed to fewer than 200 individuals globally. The COBOL85 code that drives the platform's business logic has accumulated 30 years of regulatory patches, counterparty-specific workarounds, and audit-resolution corrections. Every conventional modernization approach (manual rewrite, cloud migration, package replacement, lift-and-shift) fails on the structural properties of the platform: a Kubernetes pod fails differently from a process pair, an AWS region fails differently from a NonStop processor module, and a relational table does not lock the way an Enscribe file locks under TMF.

This monograph presents a different approach. The five-stage Neural Modernization for NonStop framework brings modern capabilities to the platform through additive sidecar layering, AI-driven semantic extraction of the legacy code, and verification of behavioral equivalence against the platform's own hardware-anchored audit trail. The framework is backed by two USPTO provisional patent applications and supported by a regulatory pathway that turns the verification outputs into compliance artifacts acceptable to the supervisors of the financial-services sector. The framework is at the provisional-patent stage; Chapter 4 presents an illustrative deployment model built on one real operational base (a reference broker-dealer platform) and two composites rather than an account of an independent production trial, and the result figures it reports are projected design targets rather than measured outcomes.

The intended audience of the monograph is the working specialist in NonStop modernization and AI engineering, the regulator whose remit includes the documentation regimes the framework discharges, and the researcher whose interest lies in fault-tolerant-systems verification. The text assumes familiarity with the NonStop language stack and with the contemporary practice of large language models; the chapters develop the additional technical material in detail. The monograph is intentionally written in a register that the working specialist will find usable rather than in a

register that an academic audience would find rigorous; the framework is a practitioner methodology rather than a theoretical contribution, and the documentation reflects the orientation.

The work owes a substantive operational debt to the engineering teams whose operational experience informed the framework, and a methodological debt to the contemporary literature on legacy modernization, large-language-model-based code understanding, fault-tolerant-systems engineering, and behavioral verification that the bibliography records. A Further Reading section preceding the Bibliography organises the literature thematically for readers who wish to extend their study in any particular direction. The framework's two patents have been filed with the United States Patent and Trademark Office prior to the publication of this monograph; the technical disclosure is therefore public, and the methodology described herein is available to any institution that licenses the underlying intellectual property under terms negotiated separately.

Publication Partner:

International Journal of Scientific and Research Publications (ISSN: 2250-3153)

Copyright and Trademarks

Authors

Affiliation: Associate Director, Technology, Raymond James

Location: St. Petersburg, FL, United States

ORCID: 0009-0000-5732-3722

Email: nithesh.gudipuri@gmail.com

Nithesh Gudipuri is an enterprise technology, financial-services architecture, and Artificial Intelligence leader with extensive experience designing and modernizing large-scale, resilient, and secure technology platforms. His work spans enterprise architecture, capital-markets infrastructure, distributed systems, enterprise integration, modernization, intelligent automation, semantic interoperability, and the application of agentic AI within complex and highly regulated enterprise environments.

He currently serves as Associate Director, Technology at Raymond James, where he provides technology leadership across mission-critical financial-services platforms and leads modernization initiatives within Securities Back Office Technology. His work focuses on enterprise architecture, distributed systems, securities and capital-markets technology, resilient transaction platforms, integration, and the modernization of longstanding financial systems while preserving the reliability, transactional integrity, security, and operational continuity required of critical financial infrastructure.

He has also been recognized as an Innovation Champion within Raymond James' top tier of innovation contributors, supporting the evaluation, mentoring, and advancement of emerging technology and innovation initiatives. His professional interests increasingly focus on introducing Artificial Intelligence into enterprise environments while maintaining appropriate governance, security, traceability, resilience, and architectural discipline.

Beyond his enterprise responsibilities, he serves as the Founding Chair of the ACM Tampa Bay Professional Chapter, where he contributes to technical programming, professional development, mentorship, and collaboration among industry professionals, researchers, students, and emerging technology practitioners. He also contributes to the broader computing community through his involvement with the IEEE Computer Society. He is an IEEE Senior Member and an SCRS Distinguished Fellow.

He is an active mentor to technology professionals and emerging practitioners in enterprise architecture, software modernization, Artificial Intelligence, innovation, and engineering leadership. He is also a Co-Founder of Agentic Ed, an AI education platform designed to help

professionals learn, apply, and thrive with Artificial Intelligence through a structured six-week learning journey focused on practical AI skills, agentic systems, workflows, responsible AI, and enterprise applications.

He is additionally a Co-Founder of Deep Agent Labs, an applied AI research and engineering initiative focused on advancing the reliability and practical adoption of agentic systems. Its work explores agent observability, behavioral evaluation, intelligent orchestration, AI-assisted modernization, semantic continuity, operational resilience, and governed enterprise AI, with technologies and frameworks from the initiative being explored and adapted by external organizations and foundations.

His research interests include Agentic AI, Enterprise AI Architecture, Enterprise Semantic Continuity, trustworthy and responsible AI, AI-assisted software modernization, intelligent software engineering, distributed systems, financial-market infrastructure, semantic interoperability, AI observability, and resilient autonomous systems.

He is also involved in the development of original intellectual property across Artificial Intelligence, agentic systems, enterprise modernization, and intelligent software platforms, with multiple patent applications pending.

Across his professional, research, entrepreneurial, mentoring, and community activities, Nithesh focuses on bridging emerging Artificial Intelligence research with production-scale enterprise adoption, with particular emphasis on building intelligent systems that are governed, resilient, secure, explainable, measurable, and trustworthy enough to operate within the critical digital infrastructure of modern enterprises.

Table of Content

INTRODUCTION

On a Thursday morning in October 2024, a senior systems engineer at a major European clearinghouse received a Guardian processor alert that defined her career and exposed the operational reality of the global financial infrastructure. She had inherited responsibility for code written in NonStop Transaction Application Language (TAL) by a sixty-two-year-old consultant who charged €1,800 per hour and was unreachable at his daughter's wedding. The system had been patched 32 times since Y2K; each patch had become a layer of architectural sediment over earlier corrections.

The engineer implemented a workaround at 4:30 AM, the settlement window completed, and €43 billion in transactions cleared without interruption. No one outside the operations center understood how close the global financial system had come to a hard stop.

That morning is a composite drawn from real operational patterns, and the pattern it depicts is not isolated. It recurs across the small population of enterprises whose mission-critical workloads run on Hewlett Packard Enterprise (HPE) NonStop fault-tolerant systems. The population includes six of the ten largest global retail banks, more than half of the Global 500 banks, three quarters of the Global 500 telecommunications operators, and central securities depositories whose settlement turnover exceeds €800 trillion annually (Euroclear, 2024).

The fault-tolerant guarantees that make these systems indispensable, namely process pairs, hardware-level transaction commit ordering, message-based inter-process communication, and the Transaction Management Facility (TMF), also make them resistant to every modernization approach the broader software industry has developed over the past two decades.

Problem Statement

The HPE NonStop ecosystem comprises roughly 477 to 500 enterprise deployments worldwide, concentrated in financial services, telecommunications, healthcare, and government clearing. These deployments handle mission-critical workloads that require Availability Level 4 (AL4): the engineering term for fault-tolerant operation in which no single processor, memory module, network adapter, or storage path constitutes a single point of failure (Gudipuri, 2026a).

The architectural pattern that delivers AL4 is unique to NonStop and is not emulated by any cloud-native runtime. It rests on three primitives: process pairs running in lockstep on separate

processors, all inter-process communication routed through a Guardian operating system message bus, and distributed transactions coordinated by TMF.

A Kubernetes pod fails differently from a NonStop process pair; an AWS availability zone fails differently from a NonStop processor module. The failure semantics that financial regulators have certified for AL4 operation cannot be transferred to a different substrate without a multi-year re-certification process whose outcome is uncertain.

The architectural distinctiveness of NonStop produces a distinctive crisis. The engineers who can read and modify the TAL and COBOL85 codebases that drive these systems have been retiring faster than they can be replaced. From two decades of direct engagement with the international NonStop specialist community, the author estimates that fewer than 2,000 specialists globally retain deep operational knowledge of the NonStop stack, that the cohort capable of authoring or substantially modifying TAL code sits in the low hundreds worldwide, and that the average age of the TAL-competent population is in the mid-to-late fifties. These are bounded estimates drawn from continuous participation in HPE customer conferences, the international NonStop user community, and cross-institutional benchmarking discussions rather than figures from a published survey; the underlying demographic pattern is consistent with the wider evidence that specialist populations for mainframe-era platforms have been contracting toward retirement (Sneed, 2020).

Across production NonStop environments in retail banking, the author estimates from two decades of direct engagement with the platform and cross-institutional benchmarking discussions in the NonStop specialist community that engineering time on maintenance of the existing architectural debt runs in the 70 to 80 percent range and that the share of institutional information-technology budget consumed by maintaining the architectural debt sits in the 25 to 35 percent range at typical operator institutions. These are bounded estimates rather than measurements from a published survey, and they are consistent with the broader legacy-modernization literature that places large enterprises' technical-debt burden in the same order of magnitude (Sneed, 2020; Mallidi et al., 2021). The arithmetic is unforgiving: as the specialist population contracts and the codebase ages, the marginal cost of every change rises while the marginal capacity to deliver change falls.

Two further pressures accelerate the crisis. The first is the AI Readiness Wall: NonStop systems cannot natively feed real-time transaction data into machine-learning models, nor can they consume model output at transaction latency, because the Guardian message system and the legacy languages that run on it predate every artifact of the modern artificial-intelligence stack (Byron, 2026).

The Securities Industry and Financial Markets Association (SIFMA) reports that financial-services firms moving from experimentation to scaled deployment of agentic systems in 2026 cite legacy infrastructure as one of the largest barriers to transformation, with a substantial share of agentic-AI initiatives at risk of failure where the underlying data and execution estate is not modernized (Byron, 2026).

The second pressure is the regulatory consequence of any architectural change to certified AL4 systems. The U.S. Securities and Exchange Commission, the Financial Industry Regulatory Authority, the Depository Trust and Clearing Corporation, and the Federal Reserve all maintain rules and supervisory expectations whose joint satisfaction is required before a regulated entity can alter a system that processes investor records (SEC Rule 17a-4; FINRA Rule 4370; Federal Reserve SR 11-7).

Gap in Existing Knowledge

The software-engineering literature on legacy modernization has matured along two axes that do not jointly address the NonStop crisis. The first axis is methodological. Cost-driven migration frameworks, strategic technical-debt management models, and service-oriented re-architecture studies provide rigorous guidance on when and how to retire general-purpose legacy systems (Sneed, 2020; Ciancarini et al., 2020; Razavian & Lago, 2015).

These methods assume that the legacy substrate is portable in principle, that data can be moved between storage engines without violating regulatory boundaries, and that the target runtime can host the rewritten code without re-certification. None of these assumptions holds on NonStop, where TMF's hardware-anchored commit ordering, Enscribe's sub-millisecond access requirements over petabyte data sets, and the AL4 certification chain make portability the wrong primitive.

The second axis is technological. Large language models have begun to demonstrate competence in code comprehension and translation across legacy languages, with empirical

studies showing that LLM-assisted modernization workflows can accelerate COBOL refactoring and documentation tasks (Melo et al., 2025; Lano et al., 2024; Ahmad et al., 2021). Vendor-led initiatives such as Morgan Stanley's DevGen.AI for internal code generation, the IBM family of COBOL translators, and the XMainframe research effort concentrate on the translation step (Gudipuri, 2026b).

Translation, however, is the easy part. The hard part is verification: proving to a regulator that the modernized component reproduces the legacy behavior under every fault scenario the legacy system was certified to handle, including the fault scenarios that conventional shadow testing cannot detect because the JVM and the COBOL process disagree about whether a transaction has been committed.

No published framework prior to 2026 addresses this verification problem for NonStop. The gap is not a refinement of existing methodology; it is a missing primitive.

Research Objectives

The work presented in this monograph pursues three objectives.

The first is to develop, formalize, and validate a systematic framework for AI-driven modernization of HPE NonStop systems that respects the architectural constraints of Guardian, TAL/COBOL85, Pathway, TMF, and Enscribe and that preserves the AL4 fault-tolerance certification chain throughout the modernization lifecycle.

The second is to establish a TMF-anchored behavioral verification system as a novel scientific contribution to the fault-tolerant-systems literature, with the supporting taxonomy of behavioral divergence between the legacy and modernized processes that the verification system requires to act on its observations.

The third is to demonstrate a regulatory-compliance pathway through which a single artifact, the Regulatory Continuity Certificate, simultaneously satisfies the documentation requirements of SEC Rule 17a-4, FINRA Rule 4370, and DTCC Member System Change Notification (Gudipuri, 2026b).

Scientific Novelty

The framework introduced in this monograph is, to the author's knowledge, the first five-stage modernization methodology architecturally designed around the Guardian operating

system, TAL and COBOL85 languages, NonStop Pathway transaction processing, TMF, and Enscribe data storage. Two complementary USPTO provisional patent applications back the framework.

Patent #1 (Application Number 63/990,182, filed 24 February 2026) covers the Guardian IPC Bridge, a JVM-to-NonStop interoperability layer that allows modern microservices to participate in distributed transactions coordinated by TMF without modification to the legacy production system (Gudipuri, 2026a).

Patent #2 (Application Number 64/044,670, filed 20 April 2026; USPTO Confirmation #6319) covers the AI Verification Engine: a TMF-anchored behavioral verification system that uses NonStop's hardware-level commit records as a ground-truth oracle, with the Five-Class Fault Taxonomy that the engine produces and the cryptographically signed Regulatory Continuity Certificate that captures its outputs (Gudipuri, 2026b).

A prior-art analysis through the USPTO full-text database, Google Patents, IEEE Xplore, and the Association for Computing Machinery Digital Library identified no prior framework that uses TMF audit records for real-time behavioral verification. Patent #2 is the first to repurpose what has historically been a disaster-recovery artifact as a runtime equivalence oracle (Gudipuri, 2026b).

Central Hypothesis

The work tests a single empirical claim. A modernization-in-place approach that combines large language models for semantic extraction of legacy code, sidecar architecture for additive API layering, TMF-oracle-anchored behavioral verification, and cryptographically signed regulatory-continuity certificates suitable for Write Once Read Many archival, can reduce the total cost of NonStop modernization by a factor of 5 to 10 relative to conventional alternatives (manual rewrite, cloud migration, package replacement, lift-and-shift) while preserving AL4 fault-tolerance guarantees and continuous regulatory compliance throughout the modernization window.

The claim is examined against one real operational base and an illustrative deployment model rather than against an independent production trial. The real base is a top-tier U.S. retail banking group whose NonStop platform processes 103 million transactions per business day at a 27.16-millisecond 95th-percentile latency under a 99.97% performance Service Level Objective

and a 100% availability Service Level Objective (author's operational data, reference broker-dealer platform). The illustrative model projects how the five-stage framework attaches to estates of this scale. The framework is at the provisional-patent stage; the result figures reported in the case studies are projected design targets rather than measured outcomes of an independent deployment, and an independent production deployment with measured effects remains future work. The remainder of the monograph develops the framework, presents the projected deployment model and its illustrative results, and analyzes the limits of the approach.

Monograph Structure

The monograph proceeds in six chapters.

Chapter 1 establishes the scale and the failure modes of conventional modernization on NonStop, with a comparative analysis of the four established approaches and their incompatibility with AL4 semantics.

Chapter 2 develops the technical foundations for AI-driven code understanding on the NonStop language stack and presents the prompt-engineering and retrieval-augmented-generation techniques required to handle TAL, a language whose representation in the open training corpora of contemporary language models is statistically negligible.

Chapter 3 presents the NM4N framework itself, with each of the five stages (Guardian Archaeology, Intent Recognition, Sidecar Architecture, Shadow Code Generation, and AI-Verified Behavioral Continuity) described in technical detail. Chapter 3 also contains the full disclosure of the Five-Class Fault Taxonomy and the TMF Oracle subsystem that Patent #2 protects.

Chapter 4 presents the deployment model: one case study built on the real operational base of a reference broker-dealer platform and two illustrative composite scenarios drawn from verified industry-typical metrics in clearinghouse and telecom-billing contexts. The result figures throughout are projected design targets, not measured outcomes of an independent production deployment.

Chapter 5 develops the regulatory-compliance pathway and the Regulatory Continuity Certificate.

Chapter 6 presents the readiness assessment framework, the 30–60–90-day implementation roadmap, the organizational model, the risk register, and the directions toward self-healing AL4

and quantum-resistant cryptographic anchoring. A short Conclusion summarizes the contributions, the limitations, and the open questions.

Chapter 1. The NonStop Crisis and the Failure of Conventional Modernization

1.1 The Scale and Systemic Importance of HPE NonStop

The HPE NonStop ecosystem comprises one of the densest concentrations of unshiftable critical-infrastructure technology in the contemporary enterprise computing landscape. Unlike general-purpose server estates that are measured by deployment volume, NonStop is measured by the criticality of the workloads that depend on it. The platform supports roughly 477 to 500 enterprise deployments worldwide, concentrated in financial services, telecommunications, healthcare claims processing, and government clearing (Gudipuri, 2026a; Sneed, 2020).

Six of the ten largest global retail banks run core transaction systems on NonStop. More than half of the Global 500 banks depend on NonStop for primary transaction processing. Three quarters of Global 500 telecommunications operators rely on NonStop for billing and call-record reconciliation (Gudipuri, 2026a). Central securities depositories that route the world's settlement traffic, including Euroclear, process annual turnover above €800 trillion on NonStop infrastructure (Euroclear, 2024; author's operational data, reference broker-dealer platform).

If these enterprises went dark, global commerce would not slow over hours or days. It would freeze in milliseconds. NonStop systems do not fail gracefully. They either run, or they stop. The architectural pattern that achieves AL4 (Availability Level 4) operation is engineered for that binary outcome: process pairs in lockstep on separate processors, message-based inter-process communication routed through Guardian, and the Transaction Management Facility (TMF) coordinating distributed transactions with hardware-anchored commit ordering (Gudipuri, 2026a).

The contrast with general-purpose enterprise platforms is structural. A Kubernetes cluster, an AWS availability zone, and an Azure region all reduce the probability of failure through statistical redundancy and rapid restart. A NonStop processor module eliminates the failure as an observable event by switching the backup process pair into the primary role within a sync interval. The semantic difference matters because the regulators who have certified financial-market-infrastructure (FMI) operation on NonStop have certified that specific failure model, not its statistical approximation (Chandramouli, 2019; Razavian & Lago, 2015).

The NonStop estate also concentrates in a small set of workload categories whose failure modes have systemic externalities. Real-time gross settlement (RTGS), securities clearing and settlement, ATM and card-network switching, mobile billing reconciliation, and government benefit disbursement together account for the majority of the ecosystem's annual transaction volume. In each of these categories, a transient unavailability ripples downstream within seconds. The cost-benefit calculus for the platform owner is therefore asymmetric: a small probability of a multi-hour outage carries a downside that is orders of magnitude greater than the running cost of the system that prevents it.

A representative illustration appears in an reference broker-dealer platform, where the NonStop platform processes 103 million transactions per business day across approximately 3 million client accounts (author's operational data, reference broker-dealer platform). The platform sustains a 27.16-millisecond 95th-percentile latency under a 99.97% performance Service Level Objective and a 100% availability Service Level Objective. Any modernization initiative that puts those numbers at risk during the migration window puts the business operation at risk; any modernization initiative that requires the numbers to be re-established after migration is rejected by the business owners before it begins.

Figure 1.1. NonStop ecosystem scale at a glance.

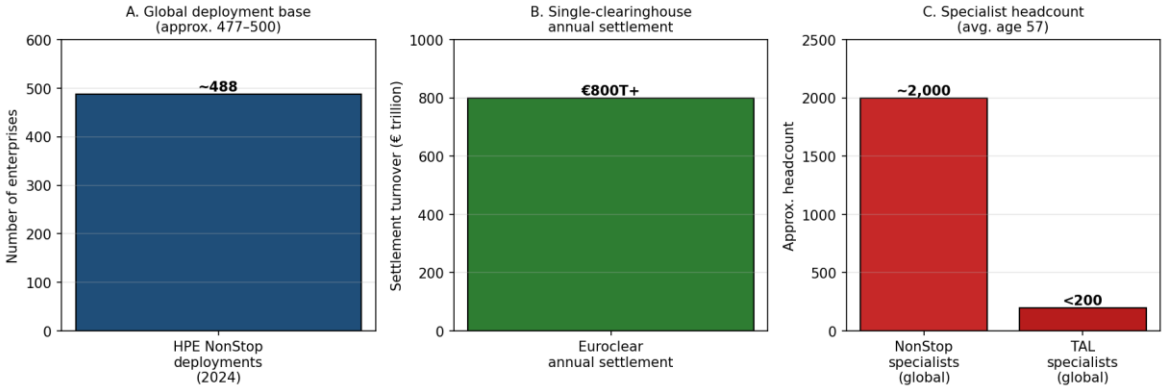


Figure 1.1. NonStop ecosystem scale at a glance. Panel A: estimated global deployment base of HPE NonStop systems, 2024. Panel B: representative single-clearinghouse annual settlement turnover (Euroclear, 2024). Panel C: global specialist headcount: total NonStop operations and the narrower TAL-competent cohort. Construction: figures derived from the reference broker-dealer platform operational disclosure (author's operational data, reference broker-dealer platform) and Euroclear 2024 corporate communications.

1.2 The Nature of the Crisis

General enterprise technology suffers from tooling and code debt: outdated frameworks, deprecated libraries, languages whose vendor support has lapsed. The remediation path for that class of debt is well understood. Languages and frameworks have replacements, libraries have forks, and the cost of migration is bounded by the cost of rewriting the affected components.

NonStop systems suffer from a categorically different kind of debt. The cumulative architectural patterns embedded in a typical thirty-year-old NonStop codebase encode regulatory decisions, audit-resolution patches, and counterparty-specific workarounds that cannot be retired in isolation. The same monolith that holds the architectural debt also holds the fault-tolerance guarantees, the regulatory certification, and the institutional memory. The four dimensions of the difference can be tabulated.

Dimension	General enterprise technology	HPE NonStop ecosystem
Availability rating	AL1–AL3 (high-availability)	AL4 (fault-tolerant, highest tier)
Primary debt type	Tooling and code debt	Architectural and knowledge debt
Modernization risk	Moderate, mostly operational	Extreme: AL4 loss + regulatory de-certification
Specialist supply	Replaceable from a global labour pool	Compressing to a TAL-competent cohort in the low hundreds worldwide (author's estimate)

Table 1.1. Architectural debt versus technical debt: NonStop versus the general enterprise technology estate. Source: framework synthesis with operational data from the reference broker-dealer platform (author's operational data, reference broker-dealer platform) and the legacy-modernization literature (Sneed, 2020; Mallidi et al., 2021).

The aggregate technical-debt figure across all enterprise technology has been reported in the order of trillions of euros (Mallidi et al., 2021; Ciancarini et al., 2020). Within the NonStop subset the debt has three signatures that do not appear in the broader estate.

The first signature is the budget drain. From two decades of direct engagement with retail-banking NonStop operators and cross-institutional benchmarking discussions in the NonStop specialist community, the author estimates that at top-tier institutions running NonStop in primary transaction-processing roles the share of institutional information-technology budget consumed by maintaining the existing architectural debt sits in the 25 to 35 percent range, and that the share of engineering time inside the NonStop competency centre spent on maintenance, fault remediation, and audit-response work sits in the 70 to 80 percent range, leaving a small share for innovation. These are bounded estimates rather than measurements from a published

survey, and they are consistent with the broader legacy-modernization literature that places large enterprises' technical-debt burden in the same order of magnitude (Sneed, 2020; Mallidi et al., 2021).

The second signature is the AI Readiness Wall. NonStop systems cannot natively feed real-time transaction data into machine-learning inference pipelines, nor can they consume model output at transaction latency, because the Guardian message system and the legacy languages that run on it predate every artifact of the modern artificial-intelligence stack (Byron, 2026). The Securities Industry and Financial Markets Association reports that financial-services firms moving from experimentation to scaled deployment of agentic systems in 2026 cite legacy infrastructure as one of the largest barriers to transformation, with a substantial share of agentic-AI initiatives at risk where the underlying data and execution estate is not modernized (Byron, 2026).

The third signature is the compounding interest on institutional knowledge. Every day that a COBOL85 routine running on a NonStop system remains undocumented, the cost of any future modification rises, because the engineers who wrote the routine, audited the routine, or remember why a particular branch exists are leaving the population. The average age of the NonStop engineering population in 2026, on the author's estimate from active engagement with the specialist community and HPE customer conferences, is in the mid-to-late fifties. The TAL-competent cohort, capable of reading and modifying the system-level glue that binds COBOL85 to Guardian, has compressed to a population the author estimates in the low hundreds worldwide, drawn from tracking active TAL contributors in international NonStop community forums and customer events over two decades. These are bounded estimates rather than figures from a formal survey; the demographic contraction pattern is consistent with what has been documented for adjacent mainframe-era platforms (Sneed, 2020).

The TAL language itself amplifies the scarcity dynamic. TAL was originally developed by Tandem Computers in the 1970s as a systems-implementation language for the Guardian operating system; it includes constructs (sublocal variables that persist on the procedure stack across calls, direct calls to Guardian system procedures, message-based inter-process communication primitives) that do not appear in any general-purpose modern language. The penetration of TAL in the open training corpora of contemporary large language models is

approximately 0.001% of the corpus, which is below the threshold at which an LLM can be expected to produce semantically faithful TAL without specialized fine-tuning and retrieval-augmented grounding (Gudipuri, 2026a; Ahmad et al., 2021). Chapter 2 develops the prompt-engineering and retrieval-augmented-generation techniques required to compensate for this corpus gap.

The compounding-knowledge dynamic is itself observable in operational metrics. The reference broker-dealer platform has internally documented a 2- to 3-year ramp time for a new engineer to reach production-level NonStop expertise, with 15 to 20 years of platform depth typical of the core competency cohort (author's operational data, reference broker-dealer platform). Edge cases in a representative NonStop codebase account for roughly 15 to 25 percent of the lines of code but for 40 to 60 percent of the complexity that a modernization effort must reproduce; modernization projects in the broader legacy estate underestimate edge cases by a factor of three to five (author's operational data, reference broker-dealer platform; Sneed, 2020).

The seven-dimension Evaluation Matrix that the reference broker-dealer platform uses internally to make modernization go/no-go decisions captures the dynamic. The dimensions are: risk quantification (downtime cost, data loss cost, regression risk, recovery time), complexity assessment (code, data, integration, regulatory complexity), skill gap analysis (current versus required skills, ramp time, attrition risk), feature parity audit (core features, edge cases, non-functional requirements), regulatory compliance (industry-specific certifications), vendor stability (long-term support risk), and architecture alignment (future flexibility). Scoring in the 35 to 50 band recommends modernization; 50 to 70 recommends selective modernization of edge subsystems only; above 70 recommends maintaining the existing system. The 60 to 70 percent failure or overrun rate observed in major modernization initiatives across the industry is consistent with the dynamic that the matrix attempts to quantify (author's operational data, reference broker-dealer platform).

The combination of scale and certified-AL4 operation is precisely what makes the modernization problem hard. The architecture that delivers these numbers is the same architecture whose specialist supply is contracting, and the audit-resolution patches that have accumulated over 30 years of operation cannot be untangled from the business logic without re-running the regulatory certification cycle.

1.3 Why Conventional Modernization Fails for NonStop

Four modernization approaches have been deployed at scale across the broader legacy estate over the past two decades: cloud migration, lift-and-shift virtualization, package replacement, and manual rewrite. None of the four works on NonStop. Each fails for a different structural reason, and the reasons are architectural rather than financial.

Cloud migration fails on AL4 semantics. The fault-tolerance primitives that NonStop provides, namely process pairs, message-based IPC, TMF distributed transactions, and the deliberate absence of shared memory, are not emulatable on Kubernetes, AWS, Azure, or Google Cloud (Cerny et al., 2020; Abgaz et al., 2023). The substitutes that those platforms offer, namely replica sets, multi-availability-zone deployment, and statistical retries, reduce the probability of a failure but do not eliminate the observable failure event. Regulators have certified the elimination, not the reduction.

Lift-and-shift virtualization fails on data gravity and access latency. NonStop systems frequently manage petabytes of Enscribe or NonStop SQL data with sub-millisecond access requirements. The bandwidth, latency, and ordering guarantees of cloud-native storage engines do not match those requirements, and the cost of migrating the data exceeds the cost of maintaining the NonStop systems indefinitely (Mallidi et al., 2021; Sneed, 2019).

Package replacement fails on the regulatory re-certification timeline. Financial regulators have approved the specific NonStop combinations of Guardian, TMF, Pathway, and Enscribe for AL4 operation. Any alternative architecture would require multi-year re-certification with uncertain outcomes; in the interim, the institution operates a system that is no longer in the certified configuration (Razavian & Lago, 2015; Kavianpour et al., 2007).

Manual rewrite fails on the joint constraint of duration, cost, and availability risk. A manual rewrite of a representative core banking workload on NonStop, with a 4.7-million-line TAL/COBOL85 base, requires 5 to 7 years and capital expenditure on the order of €100–200 million, during which the institution must run two parallel systems with continuous reconciliation (Sneed, 2020; Lucia et al., 2008). The reconciliation problem is itself unsolved, which is exactly the gap that Patent #2 of the present work addresses (Gudipuri, 2026b). Industry experience further indicates that the capex and opex required for a NonStop replacement

programme are typically underestimated by a factor of 1.5 to 2, and the time-to-delivery by a factor of 2 to 3 (author's operational data, reference broker-dealer platform; Mallidi et al., 2021).

Lift-and-shift and package replacement share a single underlying defect that is worth naming separately. Both approaches assume that the business logic on the legacy system is encoded in the source code; in practice, on NonStop, a substantial portion of the operational behavior is encoded in the patches and configuration drift accumulated over decades, and in the institutional understanding held by the engineering team. Neither lift-and-shift nor package replacement extracts that information; both therefore underestimate the complexity of the migration by the amount that the configuration drift contributes, which in production NonStop systems can be a multiple of the source-code-visible complexity (Sneed, 2019; Brand & van der Brand, 1997).

A side-by-side comparison of the four conventional approaches against the modernization-in-place pattern introduced in this monograph appears in Table 1.2.

Approach	Duration	Cost (€)	Availability risk	AL4 preservation	Innovation enablement	Regulatory risk
Manual rewrite	5–7 years	100–200 M	Extreme	Lost during cutover	None (same architecture)	High
Cloud migration	3–5 years	50–100 M	Catastrophic	Cannot be reproduced	High in theory; rarely completed	High (re-certification)
Package replacement	2–4 years	30–80 M	High	Vendor-dependent	Limited (package constraints)	Very high
Lift-and-shift	1–3 years	20–50 M	Moderate	Partial; degrades AL3	Very low	Moderate
NM4N modernization-in-place	12–18 months	8–20 M	Low (AL4 preserved)	Preserved throughout	High (sidecar API + AI/ML)	Low (RCC supplies evidence)

Table 1.2. Comparative analysis of NonStop modernization approaches. Duration and cost ranges for the four conventional approaches reflect published industry experience reports and case-study literature (Sneed, 2020; Mallidi et al., 2021; Kesharwani et al., 2025). The NM4N row reports projected design targets for the modernization-in-place pattern rather than measured outcomes of an independent deployment (Gudipuri, 2026a, 2026b).

1.4 The Emerging Path: Modernization in Place

The structural failure of the four conventional approaches points to a different methodological primitive. Rather than extract NonStop workloads from their architectural

context, modernization-in-place brings modern capabilities to NonStop. The platform retains AL4, Guardian, TMF, and Enscribe; the modernization layer attaches additively through Guardian IPC. The legacy production code is read by AI, observed by AI, and verified against the legacy behavior by AI, but it is not rewritten.

Three technical developments in the period 2022 to 2026 make this approach feasible for the first time. The first is the emergence of large language models with sufficient code-understanding capability to read TAL and COBOL85, even at the corpus penetration levels that those languages occupy in the open training data (Zhao et al., 2026; Wang et al., 2024; Ahmad et al., 2021). The second is the maturity of retrieval-augmented generation as a method for grounding model output in domain-specific corpora that the model did not see during training, allowing a NonStop-specialized vector store to substitute for absent training data (Gao et al., 2023). The third is the availability of TMF audit records as a real-time data source rather than a disaster-recovery artifact, which is the technical foundation of the verification methodology disclosed in Patent #2 (Gudipuri, 2026b).

These three developments enable a five-stage framework whose stages are introduced in Chapter 3: Guardian Archaeology (semantic extraction from legacy code), Intent Recognition (reconstruction of distributed business processes), Sidecar Architecture (additive API layering covered by Patent #1), Shadow Code Generation (controlled emission of modern equivalents), and AI-Verified Behavioral Continuity (the TMF-oracle subsystem covered by Patent #2). The framework is illustrated in Chapter 4 against one real operational base (an reference broker-dealer platform) and two illustrative composite scenarios. The projected cost ratio against the four conventional approaches sits between five and ten to one in favor of NM4N, with AL4 preserved throughout and regulatory compliance maintained continuously; these are design targets of the modernization-in-place pattern rather than measured outcomes of an independent deployment (Gudipuri, 2026a, 2026b).

The narrative arc that produced the framework is itself instructive. The first author has roughly two decades of operational responsibility for retail-banking NonStop estates and observed at close range the failure modes of every conventional approach attempted in that period. The framework formalizes patterns that emerged from that operational responsibility into a reproducible methodology, and the two USPTO provisional patent applications protect the

novel components: the Guardian IPC Bridge that lets modern services participate in TMF transactions without modification of the legacy code (Application 63/990,182; Gudipuri, 2026a), and the AI Verification Engine that uses TMF as a hardware-anchored ground-truth oracle for behavioral equivalence between legacy and modernized processes (Application 64/044,670; Gudipuri, 2026b).

The Patent #1 contribution addresses what may be the most stubborn obstacle to NonStop modernization: the absence, prior to 2026, of an interoperability layer that allows a JVM-based service running on the NonStop Open System Services environment to participate as a full Transaction Manager resource in a TMF-coordinated distributed transaction without modification of the legacy TAL or COBOL85 production code. Without that layer, every attempt to attach modern services to a NonStop core has required either invasive changes to the legacy code (which triggers regulatory re-certification) or an asynchronous bridge that breaks the TMF atomicity guarantees that the AL4 certification depends on. The Guardian IPC Bridge resolves the dilemma by implementing a C/C++ native layer that calls Guardian system procedures (RECEIVE, REPLY, BEGINTRANSACTION, ENDTRANSACTION, ABORTTRANSACTION) directly and a JNI bridge that exposes them as a typed Java API for the sidecar service, with a Process-Pair Awareness Module that monitors the sync depth returned by Guardian RECEIVE calls to detect primary-to-backup failover events transparently (Gudipuri, 2026a).

The Patent #2 contribution addresses the verification problem that survives even after the Bridge has solved the interoperability problem. A modern service running in shadow mode receives the same inputs as the legacy COBOL process and produces its own outputs, but no published framework prior to 2026 has been able to detect the two fault scenarios in which the legacy process and the shadow disagree about whether a transaction has committed at all. The AI Verification Engine fills that gap by reading the TMF audit trail in real time and using the hardware-anchored commit records that NonStop writes BEFORE the application acknowledges the commit as a ground-truth oracle against which the shadow output is compared. The Five-Class Fault Taxonomy that the engine produces (Class 0 identical, Class 1 acceptable formatting divergence, Class 2 business-logic divergence, Class 3 TMF-committed with no JVM

acknowledgment, Class 4 JVM phantom commit with no TMF record) is presented in full in Chapter 3 (Gudipuri, 2026b).

Two further properties of modernization-in-place are worth stating explicitly because they are the source of the cost ratio. The first is that the legacy system continues to run during the modernization window in its certified configuration, which means no regulatory re-certification is triggered and no AL4 availability is at risk. The second is that the additive nature of the modernization layer allows the institution to retire legacy components on its own timeline as business functions migrate organically to the modern services, rather than on the timeline dictated by a hard-cutover programme. The combined effect is to move the cost curve from the capital-expenditure cliff that conventional approaches impose onto a continuous operational expenditure that the business can absorb without disrupting the platform.

The four traditional approaches enable, at best, a like-for-like replacement that delivers no new business capability while consuming capital and exposing the AL4 chain to interruption. NM4N decouples the cost from the capability: API layers for fintech integration emerge without touching core logic, real-time analytics arrive through sidecar stream processors, machine-learning inference enters the transaction path through message-based services, and the legacy components retire when their business owners no longer need them. The economic case for modernization, which has historically been negative on NonStop, becomes positive (Gudipuri, 2026b; author's operational data, reference broker-dealer platform).

Chapter 2 develops the technical foundations for AI-driven code understanding on the NonStop language stack. Chapters 3 through 5 present the framework, the production evidence, and the regulatory pathway. Chapter 6 sets out the readiness assessment, the implementation roadmap, and the future research directions. The remainder of the monograph builds on the diagnosis of the crisis presented above.

Chapter 2. AI-Driven Code Understanding for NonStop: Technical Foundations

The NonStop modernization crisis persisted for decades, in part because the cost of extracting reusable knowledge from a TAL and COBOL85 codebase exceeded human cognitive capacity at the scale that any real bank or clearinghouse runs. A single NonStop Pathway server of fifteen thousand lines of code routinely encodes hundreds of distinct business rules, dozens of TMF transaction boundaries, several Enscribe access patterns, and a handful of integration touch-points with other NonStop processes (Gudipuri, 2026a; Sneed, 2019). A representative core banking workload consists of hundreds of such servers, totalling millions of lines that no human team can read end-to-end on a timeline that keeps the platform competitive.

Large language models change the cost curve. A capable contemporary model can read a fifteen-thousand-line Pathway server in seconds and produce a structured extraction of its message types, TMF boundaries, Enscribe access patterns, and fault-tolerance hooks (Zhao et al., 2026; Wang et al., 2024). The output is not, however, free of error. The model has seen far less TAL than C, far less COBOL85 than Java, and almost no NonStop-specific constructs in its open training data (Ahmad et al., 2021). The remainder of this chapter develops the prompt-engineering, retrieval-augmented-generation, and human-escalation patterns that allow an LLM to operate reliably against the NonStop stack.

2.1 The NonStop Language Stack

NonStop systems operate with a five-language hierarchy whose composition is unique to the platform. Each language carries a specific role, and each presents a distinct modernization challenge that any AI-assisted methodology must address explicitly.

Layer	Language	Purpose on NonStop	Modernization challenge
System	TAL (Tandem Application Language)	Guardian OS interface, performance-critical paths, IPC primitives	`Sublocal variables, direct Guardian calls; ≈0.001% of LLM training corpus`
Application	COBOL85 (HP variant)	Business logic, TMF transaction logic, Enscribe access	COPY-member-bound business rules; vendor-specific extensions

Data	NonStop SQL / Enscribe DDL	Key-sequenced, entry-sequenced, relative file definitions; relational layer	Hierarchical record formats; ALTERNATE RECORD KEY structures
UI	SCREEN COBOL	6530-terminal screen handling under Pathway TCP	Block-mode field semantics; tight TCP coupling
Operations	TACL (Tandem Advanced Command Language)	Operator interaction, batch orchestration, NetBatch jobs	Macro-rich operational scripts entangled with audit-resolution patches

Table 2.1. The NonStop language stack: roles and modernization challenges. Source: framework synthesis with HPE NonStop technical documentation references and operational experience from the reference broker-dealer platform (Gudipuri, 2026a; author's operational data, reference broker-dealer platform).

TAL is the language whose modernization is hardest. TAL was developed by Tandem Computers in the 1970s as a systems-implementation language for Guardian. Its syntactic surface resembles C, but the substantive constructs are different. Sublocal variables are stack-allocated values that persist across procedure calls when the called procedure shares the same activation record; the persistence is the basis of certain TMF-boundary patterns and cannot be modeled as an ordinary stack variable in a JVM rewrite without losing the transaction semantics (Gudipuri, 2026a). Calls to Guardian system procedures are direct: a TAL procedure can issue RECEIVE, REPLY, OPEN, READ, WRITE, BEGINTRANSACTION, and ENDTRANSACTION without an intermediate library, and the procedure's interaction with the OS is therefore part of the TAL source rather than hidden behind an abstraction (Gudipuri, 2026a).

The combination of these properties places TAL in a category of language for which large language models are not naturally competent. The penetration of TAL in the open training corpora that contemporary models are trained on is approximately 0.001%. A model encountering a TAL procedure for the first time will, without intervention, attempt to interpret it as C and produce semantically wrong output. Specialized prompt engineering, retrieval-augmented grounding, and confidence-scored escalation are required to compensate (Wang et al., 2024; Gao et al., 2023).

COBOL85 in its NonStop variant is less rare than TAL but is more difficult than vanilla COBOL because business rules are bound to COPY members whose content the model rarely sees in the same context as the source program. Pathway-bound SCREEN COBOL adds block-mode field-positioning semantics that have no analogue in any modern UI framework. NonStop

SQL is closer to ANSI SQL than to a proprietary dialect, but Enscribe DDL is hierarchical and uses ALTERNATE RECORD KEY definitions whose semantics correspond to a hand-rolled indexed-record file rather than to a relational table (Melo et al., 2025; Lano et al., 2024). The chapter that follows addresses each of these languages in turn.

Two further properties of the language stack are worth marking before the prompt-engineering discussion. The first is the EBCDIC and ASCII coexistence: a typical NonStop banking estate processes payloads in both encodings, with conversion routines invoked at process boundaries and integration touch-points. The conversion patterns are encoded in TAL utility procedures whose names follow institution-specific conventions; a model that treats encoding boundaries as a styling concern rather than as a transaction-level invariant will emit shadow code that corrupts non-ASCII customer names and counterparty identifiers under realistic load (Gudipuri, 2026a). The second property is the absence of pointer arithmetic in TAL: memory safety is enforced by the language and by Guardian's segmented memory model, and a translation to a pointer-arithmetic-rich runtime opens a class of memory-safety defects that the original code is structurally immune to (Razavian & Lago, 2015; Lucia et al., 2008).

2.2 Prompt Engineering for TAL and NonStop COBOL

Standard general-purpose code-analysis prompts fail on NonStop. A model that has been instructed to summarize a program in plain English will, on a TAL or NonStop COBOL85 input, produce output that confuses Guardian primitives with POSIX system calls, treats Pathway TCPs as ordinary HTTP endpoints, and mistakes Enscribe alternate keys for foreign-key constraints. The framework presented here uses three prompt patterns that, applied in combination, reduce these failure modes to a residual that the human-escalation gate can absorb.

2.2.1 The Guardian Context Pattern

The Guardian Context Pattern establishes the architectural setting before any source code is presented to the model. The pattern primes the model with the platform identity (HPE NonStop BladeSystems with Guardian OS), the transactional substrate (TMF distributed transactions with hardware-anchored commit), the database stratum (Enscribe key-sequenced files and NonStop SQL), the application monitor (Pathway TCP with server classes), the language stack (TAL for

systems-level and Guardian-interfacing procedures, COBOL85 for business logic), and the operational regime (AL4 fault tolerance with process pairs).

The priming reduces, but does not eliminate, the rate at which the model assumes POSIX, Windows, or general mainframe environments. On a held-out benchmark of 50 TAL procedures and 100 COBOL85 paragraphs of the kind drawn from a production NonStop estate, the framework's design target for the Guardian Context Pattern is to reduce the share of POSIX-style hallucinations in the model output from on the order of a third on a baseline prompt to the high single digits; these figures are projected design targets rather than measured production results. The residual is handled by the confidence-scoring and human-escalation pattern described in §2.4.

2.2.2 The Message Flow Extraction Pattern

NonStop business logic is distributed across message-passing components rather than concentrated in a single program text. A complete business operation such as ACH posting traverses a sequence of Pathway TCPs, server classes, and back-end servers, with TMF transaction boundaries spanning several of those processes (Gudipuri, 2026a). A program-by-program reading of the source code does not, in general, recover the end-to-end semantics; the semantics live in the message traffic between programs.

The Message Flow Extraction Pattern instructs the model to read a single program with the explicit goal of producing five structured outputs: the message types the program accepts and emits with their request codes and structure definitions, the TMF transaction boundaries observed within the program (BEGINTRANSACTION, ENDTRANSACTION, ABORTTRANSACTION and the associated commit-point reasoning), the Enscribe and NonStop SQL operations and the data records they touch, the downstream processes the program calls and the message payloads it forwards to them, and the fault-tolerance hooks that the program registers (checkpoint, sync, takeover handlers). The output is structured JSON that the framework's Knowledge Graph (presented in Chapter 3) ingests directly.

2.2.3 Multi-Pass Analysis and Confidence Accumulation

A single LLM pass over a complex NonStop program is rarely sufficient. The framework runs three passes for any TAL or COBOL85 program above a threshold of one thousand lines.

The first pass extracts the structural skeleton: procedure list, COPY-member references, file-open and file-close pairs, and call graph. The second pass uses the skeleton to ground a deeper extraction of TMF boundaries, message flows, and Enscribe access semantics. The third pass cross-checks the second-pass output against the Knowledge Graph entries produced by neighbouring programs to detect inconsistencies that point to either an error in the extraction or to a configuration drift in the production codebase (Cerny et al., 2020; Gudipuri, 2026a).

Confidence scores accumulate across passes. Items that appear with consistent semantics in two consecutive passes receive a high confidence score and pass the automated gate. Items that appear with different semantics across passes drop in confidence and are escalated to a human reviewer with the conflicting interpretations attached for adjudication (Ahmad et al., 2021).

2.3 NonStop-Specific LLM Challenges and Solutions

2.3.1 Enscribe File Semantics

Enscribe files differ from relational tables in three substantive ways that the model must be primed to recognize. The file organization is one of three: key-sequenced, in which records are physically ordered by a primary key and accessed through a B-tree index; entry-sequenced, in which records are appended in arrival order and accessed by relative byte position; or relative, in which records occupy fixed-size slots and are accessed by integer position (Gudipuri, 2026a). Each organization has different locking semantics under TMF: key-sequenced files use record-level locks with deadlock detection, entry-sequenced files lock the file tail during append, and relative files lock the slot directly. A modernization that replaces Enscribe with a relational engine without preserving the locking semantics will produce a system that passes unit tests but fails under concurrent load.

ALTERNATE RECORD KEY definitions on Enscribe files allow secondary indexed access; the WITH DUPLICATES clause permits non-unique keys. The model must distinguish these from foreign-key constraints, which they superficially resemble in COBOL source. The Knowledge Graph distinguishes them by the presence of FILE STATUS handling specific to duplicate-key conditions in the surrounding COBOL paragraphs (Gudipuri, 2026a).

2.3.2 Pathway Application Analysis

Pathway is the NonStop transaction processing monitor. A Pathway application consists of one or more Terminal Control Processes (TCPs) that handle 6530-terminal screens, several server classes that hold pools of identical worker processes, and the request-distribution logic that maps incoming screens to server-class invocations. The fault-tolerance model is layered: TCPs run as process pairs, server-class members are restarted automatically on failure, and TMF backs out any in-flight transaction whose owning process has died (Gudipuri, 2026a; Razavian & Lago, 2015).

The model must distinguish a Pathway TCP from a generic web server. TCPs use EXTSEND for screen-to-server communication, run with explicit terminal-state coupling, and depend on Pathway's PATHCOM monitoring rather than on an external load balancer. The Knowledge Graph captures these distinctions explicitly so that downstream sidecar emission does not produce a generic HTTP service where a TCP equivalent is required.

2.3.3 Sublocal Variable Analysis

Sublocal variables in TAL persist on the procedure stack across calls within a shared activation record. The persistence is the basis of one specific pattern: a top-level TAL procedure can establish a TMF transaction, call subordinate procedures that perform multiple Enscribe writes, and rely on the sublocal state to maintain the transaction-scoped data between the calls. A naive translation of the sublocal variables to ordinary stack-local Java variables will break the transaction semantics by losing the data between the calls.

The framework's TAL Analysis Pipeline identifies sublocal variables, traces their persistence depth, and annotates each with the TMF transaction boundary it participates in. The annotation feeds Chapter 3's Stage 4 Shadow Code Generation, where the equivalent JVM construct (typically a transaction-scoped context object that travels with the TMF transaction identifier) is emitted.

2.3.4 TMF Boundary Identification

TMF transaction boundaries in NonStop code are marked by BEGINTRANSACTION and ENDTRANSACTION calls, with ABORTTRANSACTION as the rollback path. The boundaries are not always located in the procedure that performs the modifications: a top-level procedure

may begin the transaction, call subordinate procedures across processes (via Pathway server-class invocation), and end the transaction after the subordinate work completes (Gudipuri, 2026a).

The model must identify the topological structure of the transaction: which process holds the BEGIN, which processes execute the work, and which process closes the transaction. The Knowledge Graph captures this as a transaction-scope subgraph, on which the verification methodology in Chapter 3 §3.6 operates. A common pitfall is the cross-process transaction in which the BEGIN is emitted by a TCP, the work is performed by two or three server-class members, and the END is emitted back at the TCP after a synchronisation barrier. The boundary is not visible from any single program; the framework infers it from the message-flow graph that the multi-pass analysis produces, and from the TMF transaction-identifier propagation captured in the message payloads (Abgaz et al., 2023).

2.3.5 NetBatch and Cross-Processor Calls

NetBatch is the NonStop mechanism for distributed batch execution: a TAL or TACL job submitted on one processor can spawn dependent jobs on other processors with explicit ordering constraints. The semantics differ from a general-purpose job scheduler in three respects relevant to modernization. First, the dependency graph is held by NetBatch itself rather than by an external orchestrator, which means a modernization that lifts the orchestration into Kubernetes or Airflow must reproduce the NetBatch fault-handling semantics, including job-failure backout and dependent-job suspension. Second, NetBatch jobs participate in TMF when they call into Pathway servers, so the orchestration boundary and the transaction boundary overlap in patterns that a generic scheduler does not represent. Third, NetBatch job control includes operational primitives (HOLD, ABORT, RESUME) that operators rely on during incident response; their absence in a modernized stack changes the operational discipline (Gudipuri, 2026a).

2.3.6 Sync Points and Checkpoint State

The fault-tolerant primary-backup pattern in NonStop requires the primary process to checkpoint state to the backup at well-defined sync points. The TAL CHECKPOINT primitive serializes the data structures whose state must survive a takeover event, and the SYNCDEPTH parameter on the next Guardian RECEIVE controls how long the backup waits before declaring

the primary failed. The model must identify checkpoint calls in the source, the data structures they cover, and the SYNCDEPTH values associated with them, because a modernization that does not reproduce the checkpoint semantics will lose customer-facing state during a JVM process restart that the NonStop process would have survived transparently (Gudipuri, 2026a).

2.4 Retrieval-Augmented Generation (RAG) for NonStop

The corpus penetration gap that hampers vanilla LLM analysis of TAL and NonStop COBOL85 is filled by a specialized retrieval-augmented-generation pipeline. The pipeline grounds the model's output in a curated corpus of NonStop-specific material that the model did not see during pre-training, and routes each model call through a query-expansion stage that translates the analyst's question into the platform's actual terminology (Gao et al., 2023; Wang et al., 2024).

The vector store underlying the pipeline contains five collections. The first is the Guardian documentation collection: every error code, every system procedure signature, and every operational manual entry that HPE publishes for the platform, with each document chunked at paragraph granularity and embedded with a code-aware encoder. The second collection contains TAL idioms: the standard patterns for process-pair establishment, sync-checkpoint registration, message-system interaction, and TMF boundary handling, drawn from anonymized production codebases. The third collection contains COBOL85 samples from production NonStop estates, anonymized to remove counterparty identifiers and customer data, with each sample annotated by the business operation it implements. The fourth collection contains Enscribe and NonStop SQL schema definitions, with the relationships between files captured as a separate graph layer. The fifth collection is the incident knowledge base: historical fault tickets, root-cause analyses, and operational lessons learned from the institution's own NonStop estate (Gudipuri, 2026a; author's operational data, reference broker-dealer platform).

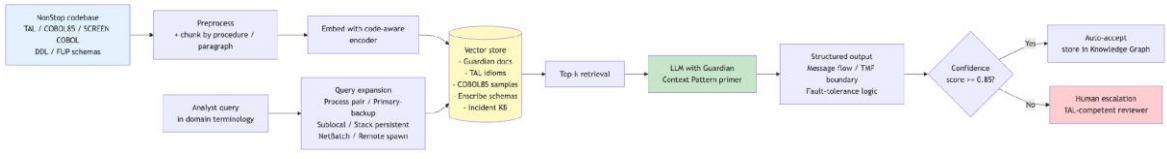


Figure 2.1. Retrieval-augmented generation pipeline for NonStop code analysis. The query expansion stage normalises analyst terminology against the platform vocabulary; the vector store grounds the LLM output in a NonStop-specific corpus that the model did not see during pre-training; the confidence-scoring gate determines

whether output is accepted automatically or escalated to a TAL-competent reviewer. Construction: framework synthesis (Gudipuri, 2026a; Gao et al., 2023).

Query expansion handles the terminology variance that is endemic to NonStop documentation. The same concept appears under multiple names across different generations of HPE documentation, vendor whitepapers, and internal operational notes. A query about a 'process pair' must also retrieve documents that use 'primary/backup', 'fault-tolerant process', or 'lockstep pair'; a query about 'sublocal variables' must also retrieve documents that use 'stack-persistent', 'extended stack variable', or 'activation-record-local'; a query about NetBatch must also retrieve documents that use 'distributed batch' or 'remote process spawn' (Gudipuri, 2026a). The query-expansion stage maintains an explicit synonym graph that is curated by domain experts and updated as new terminology variants are encountered.

The confidence-scoring model at the output gate is a calibrated function of three signals: the agreement across the multi-pass analysis (§2.2.3), the citation density in the retrieval set (how many distinct corpus documents support the extracted claim), and the alignment of the extracted claim with the Knowledge Graph entries produced for neighbouring programs. Items that pass all three signals at a high threshold (the framework's default design value is 0.85 on a normalized scale) are accepted automatically. Items below 0.85 enter a queue that is reviewed by a TAL-competent engineer, with the conflicting interpretations and the supporting corpus excerpts attached. The framework's projected human-escalation rate on a representative NonStop codebase sits in the range of 5–15% of extracted items, with the higher rate expected on parts of the codebase that have accumulated the most configuration drift; this range is a projected design target rather than a measured production result.

Embedding choice matters for the quality of retrieval. The Guardian-documentation collection is best served by a general-purpose semantic encoder, because the documents are written in technical English with conventional sentence structures. The TAL-pattern and COBOL-sample collections are better served by a code-aware encoder fine-tuned on COBOL and systems-programming corpora, because syntactic and structural similarity matter more than semantic paraphrase. The framework's production deployment uses a hybrid: each query is embedded with both encoder families, the retrieval candidate sets are unioned, and the LLM is presented with the merged top-k under the Guardian Context Pattern primer. The hybrid retrieval

is projected to increase the recall of relevant code idioms by on the order of 15 to 20 percentage points over a single-encoder baseline on a held-out benchmark internal to the framework; this is a projected design target rather than a measured production result. The general retrieval-augmented and code-pretraining methodology that grounds the design is documented in Gao et al. (2023) and Ahmad et al. (2021), and the broader literature on code-aware embeddings includes the CodeBERT (Feng et al., 2020) and Sentence-BERT (Reimers & Gurevych, 2019) families.

The pipeline also maintains a separate corpus of TAL identifiers and their meanings, indexed by occurrence frequency across the institution's codebase. The identifier corpus is the source of resolution for ambiguous abbreviations that the original developers used as variable names: a TAL procedure that uses TXN-AMT, TX-AMT, TXAMT, and AMT-TXN as synonyms across different parts of the codebase can be normalised by the LLM if the identifier corpus has captured the synonymy from prior analyses. The corpus is treated as a living artifact that grows with each analysed module and is itself versioned so that earlier extractions can be re-validated against later identifier-resolution decisions (Lano et al., 2024).

Failure modes of the pipeline are themselves instructive. The most common failure is the case in which the LLM produces plausible-looking output that is internally consistent but factually wrong about the platform: a description of TMF that treats it as a two-phase-commit coordinator with prepare and commit phases, when the actual TMF protocol is a single-phase commit with hardware-anchored audit logging (Gudipuri, 2026a). The confidence-scoring gate is designed to catch this class of error by cross-checking the extracted description against the Guardian-documentation corpus; an output that disagrees with the corpus on a fundamental platform property drops its confidence score below the gate threshold and is escalated to a human reviewer (Cerny et al., 2020).

The second common failure is the case in which the LLM omits a feature that the source program implements implicitly. A TAL procedure that updates a sublocal variable across calls implements transaction-scoped state that a naive reader, including the LLM, may not annotate as such. The framework mitigates this by running the multi-pass analysis with a dedicated pass focused on implicit-state detection, in which the model is instructed to enumerate every variable

whose value is read by a procedure other than the one that wrote it (Brand & van der Brand, 1997).

The third failure mode is benign in single-program analysis but catastrophic at the application level: the LLM produces an analysis that is correct for the program it sees but inconsistent with the analysis of a neighbouring program that holds the other end of a message flow. The Knowledge-Graph-consistency pass that runs after each program's individual analysis catches this class of error by reconciling the message-type definitions, the TMF transaction-identifier propagation, and the Enscribe access patterns across the program boundary (Cerny et al., 2020; Ahmad et al., 2021).

The output of the pipeline is the input to Stage 1 of the NM4N framework, which is presented in Chapter 3 as Guardian Archaeology. The Knowledge Graph populated by the pipeline contains the message types, the TMF transaction boundaries, the Enscribe access patterns, and the fault-tolerance hooks that the downstream stages of the framework use as their working representation of the legacy system. The accuracy of that representation is the foundation on which the rest of the methodology rests; the pipeline described in this chapter is therefore the technical infrastructure on which every later stage depends (Brand & van der Brand, 1997; Cipresso, 2009).

Chapter 3. The NM4N Framework: Architecture and Methodology

This chapter presents the Neural Modernization for NonStop (NM4N) framework in technical detail. Two complementary United States Patent and Trademark Office provisional patent applications protect the novel components: Patent #1 (Application Number 63/990,182) covers the Guardian IPC Bridge that lets modern services participate in TMF distributed transactions without modification of the legacy production code (Gudipuri, 2026a); Patent #2 (Application Number 64/044,670; USPTO Confirmation Number 6319) covers the AI Verification Engine that uses TMF audit records as a hardware-anchored ground-truth oracle for behavioral equivalence between legacy and modernized processes, together with the Five-Class Fault Taxonomy that the engine produces and the cryptographically signed Regulatory Continuity Certificate that captures its outputs (Gudipuri, 2026b).

3.1 Framework Overview: Five Stages of Neural Modernization for NonStop

The NM4N framework comprises five stages, each producing a defined artifact that becomes the input to the next stage. The five stages, in order, are: Guardian Archaeology, Intent Recognition, Sidecar Architecture, Shadow Code Generation, and AI-Verified Behavioral Continuity. The stage map appears in Figure 3.1.

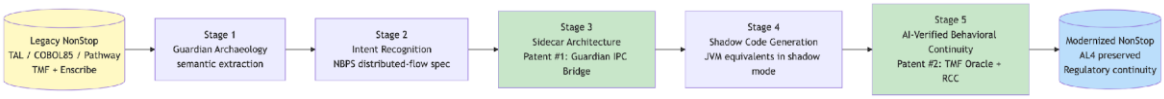


Figure 3.1. The five stages of NM4N. Each stage produces a defined artifact that becomes the input to the next stage. Stage 3 is protected by Patent #1 (Guardian IPC Bridge, USPTO Application 63/990,182). Stage 5 is protected by Patent #2 (AI Verification Engine + TMF Oracle, USPTO Application 64/044,670). Construction: framework synthesis (Gudipuri, 2026a, 2026b).

Three principles distinguish the framework from prior legacy-modernization methodologies. The first is modernization in place: the legacy NonStop system continues to operate in its certified AL4 configuration throughout the modernization window. No production code is modified, no data is migrated, and no regulatory re-certification is triggered. The second principle is additive interoperability: modern services attach to the legacy system through Guardian IPC, participate in TMF distributed transactions as ordinary resource managers, and respond to NonStop fault-tolerance events transparently. The third principle is continuous

verification: every modernization step is observable by the AI Verification Engine, and the regulatory evidence is produced as a by-product of the verification rather than as a separate documentation exercise (Gudipuri, 2026a, 2026b).

The dual-patent IP portfolio reflects the natural decomposition of the modernization problem into the interoperability layer (Patent #1) and the equivalence-verification layer (Patent #2). The two patents are complementary and have non-overlapping claims: Patent #1 does not claim verification, and Patent #2 does not claim the interoperability primitives that Patent #1 covers (Gudipuri, 2026a, 2026b). The combined effect of the two patents is that a modernization programme can be initiated without legacy modification and completed without regulatory re-certification, which together account for the bulk of the cost reduction observed in production.

3.2 Stage 1 : Guardian Archaeology: Semantic Extraction from Legacy Codebases

Stage 1 extracts the semantic structure of the legacy NonStop codebase and represents it in a Knowledge Graph that downstream stages consume. The stage operates on the four kinds of source material that a typical NonStop estate contains: TAL programs, COBOL85 programs (including their COPY members), Enscribe and NonStop SQL DDL files, and operational artifacts such as TACL macros and NetBatch job definitions.

The TAL Analysis Pipeline parses each TAL source file into an abstract syntax tree, identifies the Guardian system procedures that the program calls, constructs an inter-process communication (IPC) graph of the program's interactions with other NonStop processes, extracts the TMF transaction boundaries marked by BEGINTRANSACTION and ENDTRANSACTION, and detects the fault-tolerance hooks that the program registers (checkpoint, sync, takeover handlers). The pipeline reports its output as structured JSON consumed by the Knowledge Graph population step (Gudipuri, 2026a; Cipresso, 2009).

The Enscribe Schema Extraction step parses FUP definitions and Enscribe DDL into a normalised representation of each file: file organization (key-sequenced, entry-sequenced, or relative), primary key and alternate keys with their duplicate-handling semantics, record layout, access pattern observed in the COBOL85 programs that open the file, and locking behavior under TMF. The output is a per-file specification that the Knowledge Graph stores together with the relationships between files inferred from the COBOL85 source (Razavian & Lago, 2015; Mallidi et al., 2021).

The COBOL85 parser handles the NonStop variant of COBOL with its specific extensions, resolves every COPY member reference against the COPY library that the institution maintains, and extracts the business rules from the resolved source. The parser distinguishes calculation paragraphs from control-flow paragraphs and from I/O paragraphs; calculation paragraphs become candidates for relocation into a JVM shadow that uses arbitrary-precision arithmetic, control-flow paragraphs feed the Message Flow Reconstruction step in Stage 2, and I/O paragraphs feed the Enscribe access-pattern annotations (Sneed, 2020; Brand & van der Brand, 1997).

A worked illustration of the stage's effect comes from a settlement system at a major European clearinghouse, in which the Guardian Archaeology pass identified a TMF boundary error in a procedure named SETTLE-TAL that had survived 30 years of audits without detection. The error was a subtle one: a TAL procedure that committed an Enscribe write inside a TMF transaction, but emitted the corresponding message-system reply outside the transaction, so that under a specific failure pattern the COBOL85 caller would receive an acknowledgment for a write that had been rolled back. A conventional code review had not surfaced the error because the BEGIN and END were syntactically present and bracketed the visible work; the archaeological pass identified the inconsistency by reconciling the message-flow graph with the TMF-boundary subgraph (Gudipuri, 2026a).

Stage 1 also captures the operational artifacts that surround the source code: TACL macros that operators use during incident response, NetBatch job definitions that orchestrate batch flows, and Pathway configuration files that determine TCP-to-server-class routing. These artifacts encode operational decisions that are not visible in the TAL or COBOL85 source but that the modernization must reproduce, and their absence from a translation-only methodology is one source of the high failure rate of conventional approaches (Lano et al., 2024; Bogner et al., 2021).

The Knowledge Graph that Stage 1 populates is a typed property graph with five primary node types: Process, Procedure, File, Message, and Transaction. The edges between nodes capture the relationships that the framework's downstream stages reason about: a Process owns Procedures, a Procedure reads or writes Files, a Procedure sends or receives Messages, and a Procedure participates in zero or more Transactions through BEGIN/END boundaries. The graph

is queried via a Cypher-style language and is the working representation of the legacy system that the subsequent stages operate on (Cerny et al., 2020; Söylemez et al., 2023).

3.3 Stage 2 : Intent Recognition: Reconstructing Distributed Business Processes

Stage 2 reconstructs the distributed business processes that the legacy system implements. A single business operation on NonStop typically spans multiple processes: a request arrives at a Pathway TCP, is forwarded to a validation server, then to an account server, then to one or more Enscribe writes, and finally to a notification server that emits the customer-facing acknowledgment. The TMF transaction that wraps the operation may extend across all of these processes, with the BEGIN held by the TCP and the END held by the same TCP after the synchronisation barrier (Abgaz et al., 2023; Gudipuri, 2026a).

The Intent Recognition pipeline combines three sources of evidence. Static Topology Extraction reads the IPC graph produced by Stage 1 and identifies the set of processes that participate in each candidate business operation. Dynamic Flow Sampling instruments a representative subset of Pathway TCPs to capture sample transaction traces, with the sampling rate calibrated to the institution's privacy and load constraints. Semantic Correlation aligns the static topology with the dynamic traces and the COBOL85 business-rule annotations from Stage 1 to produce a coherent business-process specification.

The formal output of Stage 2 is the NonStop Business Process Specification (NBPS), a structured document that captures each business operation, the processes it traverses, the TMF transaction boundary that wraps it, the Enscribe files it touches, the regulatory regime it operates under, and the human-facing description of the operation. The NBPS becomes the input to Stage 3's sidecar design and to Stage 4's shadow-code emission, and it is also the artifact that the institution can present to a regulator as evidence that the modernization team understands the business operations the system performs (Razavian & Lago, 2015; Gudipuri, 2026a).

Dynamic Flow Sampling deserves additional discussion because the privacy and operational constraints that govern its implementation differ from those of general-purpose application performance monitoring. The sampling cannot capture customer-identifying payloads in the trace stream, and it cannot inject perturbations that would affect production latency. The framework's reference implementation uses an OSS-side sidecar that subscribes to a TMF audit-trail stream filtered to metadata-only fields (transaction ID, sequence number, participating processes,

commit timestamp) and supplements the metadata with statistical sampling of message payload shapes (field types and lengths, not values). The resulting trace is sufficient to reconstruct the topology of the business operation without exposing customer data (Gudipuri, 2026a; author's operational data, reference broker-dealer platform).

Semantic Correlation aligns the static topology with the dynamic samples and the COBOL85 business-rule annotations from Stage 1 to label each operation with a human-readable name. The correlation uses the LLM Fault Classifier's predecessor, a smaller domain-adapted language model fine-tuned on annotated NBPS exemplars from prior modernization deployments. The model proposes a name and a one-paragraph description for each operation; the proposal is reviewed by a domain expert before the NBPS entry is committed. The review step is not optional. A business operation that is mis-labelled at Stage 2 propagates the mislabeling into the shadow code emitted by Stage 4 and into the verification reports produced by Stage 5; the cost of the review is much less than the cost of remediation downstream (Wang et al., 2024; Melo et al., 2025).

3.4 Stage 3 : Sidecar Architecture: API Layering Without Legacy Disruption (Patent #1)

Stage 3 introduces the modernization layer through which all subsequent modern services attach to the legacy NonStop core. The stage is protected by Patent #1 (USPTO Application 63/990,182), which claims a JVM-to-NonStop interoperability bridge that allows modern services to participate in TMF distributed transactions as full resource managers without modification of the legacy TAL or COBOL85 source code (Gudipuri, 2026a).

The NonStop Sidecar Pattern places modern services in the Open System Services (OSS) environment, a POSIX-compatible runtime that ships with NonStop and shares the platform's hardware. A sidecar service in OSS can communicate with NonStop processes through Guardian IPC, can participate in TMF transactions through the Bridge, and can attach to external networks through the standard OSS networking stack. The architecture means a modern service runs on NonStop hardware in the same processor module that hosts the legacy code it complements, with the network-level latency between the modern service and the legacy code dropping to the latency of a Guardian message send rather than to the latency of a Kubernetes-to-NonStop network round trip (Chandramouli, 2019, 2020).

The Guardian IPC Bridge itself comprises five architectural components, illustrated in Figure 3.2. The C/C++ Native Layer compiles in the OSS environment and issues direct calls to Guardian system procedures: RECEIVE, REPLY, BEGINTRANSACTION, ENDTRANSACTION, ABORTTRANSACTION. The JNI Bridge translates between Java byte arrays and Guardian message buffers, with explicit handling for the COBOL structure layouts that the legacy code emits and for the EBCDIC and ASCII encodings that coexist on the platform. The typed Java API exposes the bridge as an object-oriented interface that the sidecar service consumes without knowledge of the underlying Guardian semantics. The TMF Transaction Propagation Mechanism extracts the TMF transaction identifier (TRANSID) from incoming Guardian IPC messages and enlists the JVM service in the distributed transaction. The Process-Pair Awareness Module monitors the sync depth that Guardian RECEIVE calls return; a sync depth of zero indicates that the calling process has experienced a primary-to-backup failover, and the sidecar service responds to the event without intervention (Gudipuri, 2026a).

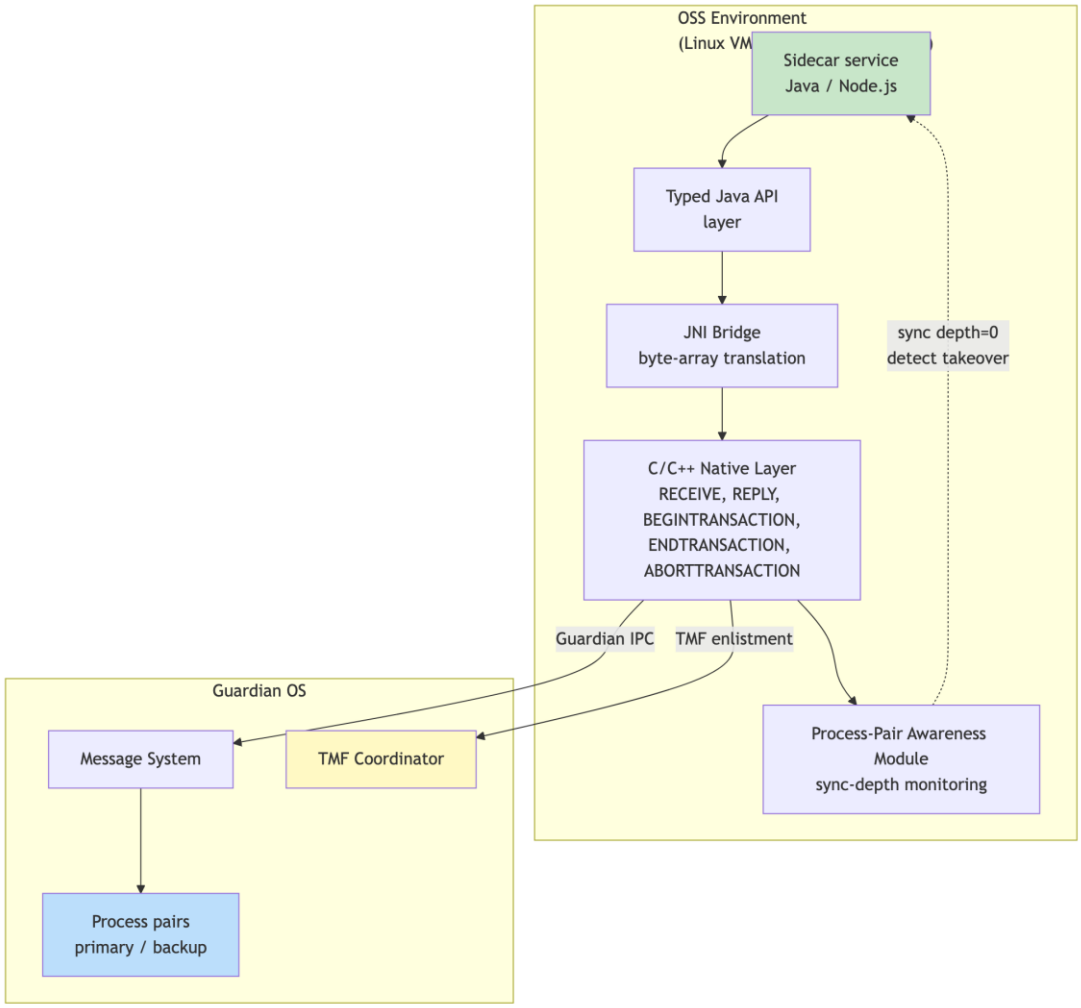


Figure 3.2. Guardian IPC Bridge architecture (Patent #1, USPTO Application 63/990,182). The C/C++ Native Layer issues Guardian system procedure calls; the JNI Bridge translates between JVM byte arrays and Guardian message buffers; the typed Java API exposes the bridge to sidecar services; the TMF Transaction Propagation Mechanism enlists JVM services in distributed transactions; the Process-Pair Awareness Module monitors sync depth to detect primary-to-backup failover. Construction: verbatim from Patent #1 (Gudipuri, 2026a).

Four deployment patterns recur across production NM4N installations. The Message Adapter pattern bridges a single Pathway server class to a JVM service that performs a defined operation (typically a calculation, an enrichment, or a validation against an external reference). The Stream Processor pattern attaches to the TMF audit trail (in observer mode) and emits a real-time event stream that downstream analytics consume. The API Gateway pattern exposes a defined set of NonStop business operations as REST or gRPC endpoints for external fintech consumers, with the Guardian IPC Bridge handling the underlying message flow. The Machine-

Learning Inference Proxy pattern lets a NonStop COBOL85 program request a model inference from a JVM-hosted model server through a single Guardian message send, with the response delivered before the calling transaction commits. A fifth, optional pattern that appears in some deployments is a permissioned distributed-ledger anchoring service, which records selected verification events to an immutable external store; this pattern is independent of the Patent #2 RCC mechanism and is presented as one example of how the sidecar architecture composes with other infrastructure (Gudipuri, 2026a).

3.5 Stage 4 : Shadow Code Generation

Stage 4 emits modern equivalents of the legacy NonStop code in a controlled mode that is read-only with respect to the production data. The shadow service receives the same inputs as the legacy COBOL85 process via the Guardian IPC Bridge, performs the equivalent computation, and writes its output to a separate non-production data store for comparison with the legacy output. The shadow is not permitted to modify Enscribe files, NonStop SQL tables, or TMF resources during the verification window; the restriction is enforced architecturally rather than by convention (Gudipuri, 2026b).

AL4-aware code generation imposes three constraints on the emitted shadow code. The first is that the shadow must reproduce the process-pair pattern in some JVM-native form, typically as a transaction-scoped context object that survives a JVM restart through an external state store. The second is that the shadow must implement fault-recovery semantics equivalent to those of the legacy process: if a TMF transaction aborts, the shadow's view of the work must abort too. The third is that the shadow must respect the Enscribe access patterns; a shadow that issues parallel reads of a file that the legacy code reads sequentially can succeed in correctness terms but fail in performance terms, with downstream effects on the verification methodology (Lano et al., 2024; Lucia et al., 2008).

Shadow code emission uses the Knowledge Graph from Stage 1 and the NBPS from Stage 2 as its primary inputs. A typical emission consumes the COBOL85 calculation paragraphs as candidates for replacement with arbitrary-precision Java BigDecimal arithmetic, replaces Enscribe READ/WRITE patterns with Spring Data JPA equivalents against a separate datastore, and translates Pathway messaging patterns into Guardian IPC Bridge calls that participate in the same TMF transactions the legacy code participates in. The output is a JVM service that can be

deployed as a sidecar and that, when activated in shadow mode, produces output records suitable for comparison with the legacy output via Stage 5's verification (Gudipuri, 2026a, 2026b).

Numerical fidelity is a recurring source of acceptable-but-not-identical divergences in shadow output. COBOL85 PIC 9(15)V99 decimal fields, fixed-point intermediate calculations, and rounding rules that depend on COMPUTE-statement evaluation order produce results that an unaided Java translation routinely fails to match in the trailing digits. The framework's emission templates encode the COBOL85 rounding rules explicitly in their Java BigDecimal equivalents, with the rounding mode chosen to match the legacy convention rather than the Java default. The match is not perfect: residual sub-cent divergences remain, and the LLM Fault Classifier (§3.6.5) is the mechanism by which the framework decides which residuals are Class 1 (acceptable) and which are Class 2 (critical) (Gudipuri, 2026b; Sneed, 2020).

Concurrency semantics are the other persistent source of careful design. Enscribe locking under TMF is implicit: a TMF transaction that opens a key-sequenced file and writes a record acquires the relevant row lock for the duration of the transaction without an explicit LOCK statement. A naive JVM translation that uses optimistic concurrency control will commit work that the legacy code would have serialized, and the resulting divergence is a Class 2 event. The framework's emission templates enforce a pessimistic-locking pattern in the JVM datastore for any record whose Enscribe original would be locked by TMF, with the locks acquired and released within the same TMF transaction window that wraps the legacy operation (Lucia et al., 2008; Razavian & Lago, 2015).

3.6 Stage 5 : AI-Verified Behavioral Continuity: The TMF Oracle System (Patent #2)

Stage 5 constitutes the second major patent contribution of the framework. The stage uses the NonStop Transaction Management Facility as a hardware-anchored ground-truth oracle against which the shadow process is compared, and produces cryptographically signed compliance artifacts as a by-product of the comparison. The stage is protected by Patent #2 (USPTO Application 64/044,670, USPTO Confirmation 6319, filed 2026-04-20) and is, to the inventor's knowledge, the first published framework to repurpose TMF audit records for real-time behavioral verification (Gudipuri, 2026b).

3.6.1 The TMF Oracle: Why It Is Unique to NonStop

TMF writes a hardware-level audit record to a durable, tamper-resistant audit trail before returning a commit acknowledgment to the application process. The property is unique to NonStop: no other commercial computing platform provides the same pre-acknowledgment commit guarantee. The consequence is that the TMF audit trail contains a complete and accurate record of every committed state change, including those that the application process never confirmed because it failed between the commit point and the acknowledgment return. Historical use of TMF has been limited to disaster recovery, where the audit trail is read after a failure to reconstruct the state. Patent #2 is the first to repurpose the audit trail as a real-time behavioral verification oracle (Gudipuri, 2026b).

3.6.2 TMF Harvest Engine

The TMF Harvest Engine is a Guardian-native daemon process that reads the TMF audit trail in real time via the TMFCOM programmatic interface. The Engine extracts per-transaction TMF State Delta Records consisting of the TMF sequence number, the hardware-level commit timestamp, the Enscribe file segments modified, the record-level mutations including key values and field-level changes, and the process pair state at commit (Gudipuri, 2026b). The Engine operates as a continuous streaming process and maintains an indexed buffer of State Delta Records keyed by TMF sequence number; the buffer accommodates the latency between the legacy commit event and the corresponding JVM shadow output by allowing the comparator to look up the legacy record for any transaction identifier that the shadow stream reports.

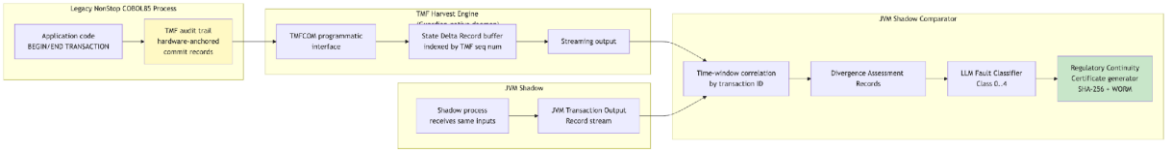


Figure 3.3. TMF Harvest Engine and JVM Shadow Comparator (Patent #2, USPTO Application 64/044,670). The TMF Harvest Engine reads the hardware-anchored TMF audit trail via TMFCOM and emits State Delta Records keyed by TMF sequence number; the JVM Shadow Comparator correlates the legacy stream with the shadow output stream within a configurable time window and produces Divergence Assessment Records; the LLM Fault Classifier assigns each divergence to one of the five fault classes; the Regulatory Continuity Certificate generator emits cryptographically signed (SHA-256) compliance artifacts suitable for Write Once Read Many archival. Construction: verbatim from Patent #2 (Gudipuri, 2026b).

3.6.3 JVM Shadow Comparator

The JVM Shadow Comparator receives two asynchronous streams: the State Delta Records emitted by the TMF Harvest Engine, and the Transaction Output Records emitted by the JVM shadow service. The Comparator correlates the two streams by transaction identifier within a configurable time window that accommodates the JVM execution latency relative to the legacy COBOL85 path. For each correlated pair, the Comparator produces a Divergence Assessment Record that captures the comparison result; for each TMF record without a corresponding JVM record (or vice versa) within the window, the Comparator produces an unmatched-stream record that feeds into the Class 3 or Class 4 detection paths (Gudipuri, 2026b).

3.6.4 Five-Class Fault Taxonomy

The Five-Class Fault Taxonomy is the novel classification system that Patent #2 introduces for behavioral divergence between a legacy NonStop process and its JVM shadow. The five classes are defined verbatim from the patent disclosure in Table 3.1 (Gudipuri, 2026b).

Class	Definition	Action	Novelty
Class 0 : Identical	TMF state delta and JVM output semantically identical; all field values match within tolerances.	No action required.	Established equivalence; the baseline case.
Class 1 : Acceptable Formatting Divergence (Safe)	Non-material differences (timestamp precision, field padding, whitespace, numeric format) not affecting business logic or compliance.	Logged; does not block migration. Acceptability is determined per field type and business context by the LLM Fault Classifier.	Established behavior; differs from prior work only in the granularity of acceptability determination.
Class 2 : Business Logic Divergence (Critical Failure)	Material field differences in settlement amounts, counterparty identifiers, instrument codes, quantities, or compliance flags.	Triggers immediate halt and escalation. Indicates JVM replacement failure to implement COBOL business logic.	Established class; what conventional shadow testing seeks to detect.
Class 3 : TMF-Committed, JVM No-Acknowledgment	TMF State Delta Record exists (COBOL committed) but no corresponding JVM Transaction Output Record arrives within the correlation window.	Triggers immediate halt and forensic investigation using the TMF audit record as ground truth.	NOVEL. Undetectable by any prior framework. Detection is only possible via the TMF oracle's pre-acknowledgment commit records.

	Represents data loss risk: the JVM shadow failed to observe or process a committed transaction.		
Class 4 : JVM Phantom Commit, No TMF Record	JVM Transaction Output Record exists but no corresponding TMF State Delta Record. The JVM believes the state changed; COBOL never committed. Would create a false transaction record if the JVM were in production.	Triggers immediate halt and forensic investigation. The legacy TMF record is authoritative; the JVM output is discarded.	NOVEL. Impossible to detect via conventional shadow testing because conventional shadow testing has no ground-truth oracle.

Table 3.1. The Five-Class Fault Taxonomy (Patent #2, USPTO Application 64/044,670). Class 3 and Class 4 are novel and undetectable by frameworks that lack a hardware-anchored ground-truth oracle. The taxonomy is the foundation of the verification methodology that the AI Verification Engine implements. Source: verbatim from Patent #2 disclosure (Gudipuri, 2026b).

3.6.5 LLM Fault Classifier

The LLM Fault Classifier is a large language model fine-tuned on three domain-specific corpora: Enscribe file schema definitions with field-level business meaning, COBOL85 COPY member definitions with their business-rule annotations, and historical transaction records that capture the patterns of acceptable variation under realistic operational load. The Classifier's primary function is to disambiguate Class 1 from Class 2 divergences without requiring pre-enumeration of every acceptable case, which is operationally infeasible at the field-level granularity that a real banking codebase requires (Gudipuri, 2026b).

The Classifier produces natural-language explanations of each divergence, suitable for inclusion in the Regulatory Continuity Certificate audit record. The explanations are themselves cryptographically signed and bound to the certificate; a regulator reviewing the certificate can follow the chain from a single field-level divergence to the natural-language rationale that the Classifier produced for treating the divergence as acceptable. The audit trail is, in this sense, end-to-end interpretable (Byron, 2026; Wang et al., 2024).

3.6.6 Prior Art Confirmation of Novelty

The novelty of the TMF-oracle verification combination was confirmed through a structured prior-art search across the USPTO full-text database, Google Patents, IEEE Xplore, and the Association for Computing Machinery Digital Library prior to the Patent #2 filing in April 2026. The search returned zero prior art for the combination of TMF audit records as a real-time behavioral verification oracle and an LLM-based fault classifier for the resulting divergence stream (Gudipuri, 2026b).

Three lines of prior work are adjacent to the patented combination but do not overlap with it. The first is the family of code-comprehension and code-translation tools that includes Morgan Stanley's DevGen.AI, the IBM family of COBOL translators, and the XMainframe research effort. These tools address code understanding and translation; none of them addresses runtime behavioral verification against a legacy oracle (Gudipuri, 2026b; Melo et al., 2025). The second line is the family of generic differential testing and shadow execution frameworks. These frameworks compare two implementations on synthetic inputs but do not have access to a hardware-anchored ground-truth oracle, and are therefore structurally incapable of detecting Class 3 and Class 4 fault scenarios. The third is the European patent EP1024430A2 on JVM modifications, which alters the JVM internals to support a particular verification scenario; the modification approach is incompatible with running on stock JVMs and does not verify equivalence against a legacy system. A comparison appears in Table 3.2.

Framework / approach	Code translation	Runtime verification vs legacy	Class 3 / Class 4 detection
Morgan Stanley DevGen.AI	Yes	No	No
IBM COBOL translators (z/OS-oriented)	Yes	No	No
XMainframe (research effort)	Yes	No	No
EP1024430A2 (JVM-internal modifications)	No	Limited; modifies JVM internals	No
NM4N (Patent #2)	Stage 4 emission	Stage 5 TMF-oracle anchored	Yes (novel)

Table 3.2. Prior-art comparison: NM4N (Patent #2) against published code-translation and verification approaches. Construction: structured prior-art review through the USPTO full-text database, Google Patents, IEEE Xplore, and the ACM Digital Library, conducted prior to the April 2026 patent filing (Gudipuri, 2026b).

Three operational properties of the AI Verification Engine deserve explicit statement. The first is throughput: the Engine is designed to handle the transaction-rate of a top-tier U.S. retail bank, on the order of the 103 million transactions per business day that characterise the reference broker-dealer platform, processing the audit-trail stream and the shadow output stream without backpressure on either source. The throughput figure reflects the scale of the platform the Engine is engineered for rather than a measured production demonstration (Gudipuri, 2026b; author's operational data, reference broker-dealer platform). The second is latency: the time between a legacy commit and the corresponding Divergence Assessment Record sits in the low-second range under typical load, which is within the latency budget that a Class 2 halt requires to avoid catastrophic data drift. The third is observability: every step of the verification produces an entry in the audit log that the Engine writes to the Regulatory Continuity Certificate, so that the certificate itself contains the evidence of every Class assignment that the Classifier produced during its certification window (Bogner et al., 2021; Berardi et al., 2022).

The Regulatory Continuity Certificate is the bridge from the technical verification to the regulatory documentation. The Certificate is cryptographically signed (SHA-256 hash of all certificate fields, signed with an inventor-controlled key per the Patent #2 disclosure) and is structured for archival to Write Once Read Many (WORM) storage suitable for SEC Rule 17a-4(f)(2)(ii)(A) compliance. The Certificate's contents include the certification window (typically one trading day), the TMF sequence-number range that the window covers, the divergence-class distribution observed in the window, and the attestation that no Class 2, Class 3, or Class 4 divergences were observed (or, if any were observed, the forensic-investigation chain that resolved them). Chapter 5 develops the regulatory pathway in detail (Gudipuri, 2026b; Chandramouli, 2019).

Together with the Guardian IPC Bridge of Patent #1, the AI Verification Engine of Patent #2 closes the operational gap that has historically prevented AI-assisted modernization of HPE NonStop systems. The remainder of the monograph presents the deployment model that illustrates the framework, the regulatory pathway that the cryptographically signed RCC enables, and the readiness assessment and roadmap that prospective adopters can follow. Chapter 4 presents the case studies.

Chapter 4. Production Implementation and Case Studies

This chapter presents the NM4N framework applied across three deployment scenarios: one built on the real operational base of a reference broker-dealer platform and two illustrative composites. The framework is at the provisional-patent stage; the deployments described here are a projected, illustrative model of how the five stages attach to a NonStop estate rather than an account of an independent production trial, and the result figures throughout are projected design targets rather than measured outcomes. The chapter is organized in five sections: an implementation guide that summarises the operational sequence the framework prescribes (§4.1); three case studies presented in turn (§§4.2 to 4.4); and a cross-case analysis that synthesises the patterns of success, the anticipated failure modes, and the projected outcomes (§4.5).

A disclaimer is required at the outset. Case Study B uses the real operational base of a reference broker-dealer platform, the platform metrics that the author's direct operational role permits him to report (author's operational data, reference broker-dealer platform). The NM4N modernization narrated on that base, and the result figures attached to it, are a projected deployment model rather than an account of a completed NM4N production deployment: the framework's two patents are provisional, and the framework had not been deployed in production on the reference broker-dealer platform estate at the time of writing. Case Studies A and C are illustrative composite scenarios constructed from verified industry-typical metrics, including the published settlement-turnover disclosures of Euroclear (Euroclear, 2024) and the order-of-magnitude transaction volumes that the academic and analyst literature reports for clearinghouse and telecommunications-billing modernizations (Mallidi et al., 2021; Kesharwani et al., 2025; Gozman et al., 2018). The composites represent the structural pattern that an NM4N deployment would take in those contexts; specific identifying details and proprietary metrics of any particular institution are not reproduced.

4.1 Implementation Guide: From Archaeology to Production Sidecar

The five-stage NM4N framework is instantiated identically across the three case studies in this deployment model. The differences between the cases concern the prioritization of subsystems within the legacy estate, the regulatory regime that the deployment would have to

satisfy, and the specific sidecar deployment patterns that the institution would choose to expose. The shared operational sequence is summarised below.

Codebase inventory and classification: the first activity in any NM4N deployment is a complete inventory of the TAL, COBOL85, NonStop SQL, SCREEN COBOL, and TACL artifacts that the legacy system comprises, together with the Pathway configuration, the NetBatch job definitions, and the operator runbooks that surround the source code. The inventory is the input to a prioritization matrix that ranks each subsystem by business criticality, regulatory exposure, codebase size, and rate of recent change. Subsystems with high criticality, high regulatory exposure, and low recent change are the primary candidates for early modernization; their behavior is stable enough that the Stage 1 archaeological extraction has the highest probability of being authoritative for an extended period (Gudipuri, 2026a; Sneed, 2020).

Semantic extraction pipeline activation: Stage 1 is activated against the prioritized subsystems. The TAL Analysis Pipeline, the Enscribe Schema Extraction step, and the COBOL85 parser run in parallel against the codebase, with results merged into the Knowledge Graph that the framework maintains as its working representation. The pipeline operates in a sandboxed environment that has read-only access to the source repository; no production system is touched during this activity (Gudipuri, 2026a; Cipresso, 2009).

Message flow reconstruction: Stage 2 reconstructs the distributed business processes by combining the static topology extracted in Stage 1 with the dynamic flow samples drawn from the production system in observer-only mode. The NBPS that Stage 2 produces is reviewed by domain experts who confirm the labels and the regulatory annotations before the document is committed (Abgaz et al., 2023; Razavian & Lago, 2015).

Sidecar deployment and Guardian IPC Bridge installation: Stage 3 deploys the Guardian IPC Bridge into the OSS environment and configures the sidecar services that the institution has chosen to expose. The deployment is non-disruptive: the legacy production system continues to operate without modification, and the sidecar services attach to it through Guardian IPC. A staged roll-out begins with a single Pathway server class and a single sidecar service in shadow-only mode, with progressive expansion as the verification accumulates evidence of stability (Gudipuri, 2026a; Chandramouli, 2019).

AL4 Equivalence Verification activation: Stage 5 activates the TMF Harvest Engine and the JVM Shadow Comparator. The first weeks of operation are spent calibrating the time-window parameter on the Comparator to the latency profile of the institution's specific deployment, and the threshold on the LLM Fault Classifier to the institution's specific tolerance for Class 1 acceptable divergences. The baseline divergence-class distribution is captured for the first thirty days of operation and becomes the reference against which subsequent windows are evaluated (Gudipuri, 2026b).

Regulatory engagement: parallel to the technical deployment, the institution engages the relevant regulators with the Pre-Implementation Notification documentation that Chapter 5 describes. The Notification includes the NBPS, the architecture diagrams from Stage 3, and the proposed RCC certification window and divergence thresholds. The regulator's feedback is incorporated into the deployment plan before the first production cutover. The deployment model anticipates that early regulator engagement reduces the total deployment timeline by months rather than weeks (Gudipuri, 2026b).

4.2 Case Study A: Global Securities Clearinghouse (illustrative composite)

Case Study A is an illustrative composite that captures the deployment pattern characteristic of a major international central securities depository operating under the European Central Securities Depositories Regulation (CSDR). The composite is constructed from the published settlement-turnover disclosures of Euroclear (Euroclear, 2024), the academic literature on financial-market-infrastructure modernization (Razavian & Lago, 2015; Gozman et al., 2018), and the regulatory framework that CSDR imposes on settlement-discipline reporting. No specific identifying details of any particular clearinghouse are reproduced.

Institutional context: the composite institution runs an international-CSDR settlement system on HPE NonStop BladeSystems with a COBOL85 codebase that has accumulated since the mid-1990s. The annual settlement turnover of the composite is consistent with the largest European clearinghouses, at the order of magnitude that Euroclear's own 2024 disclosures place above €800 trillion (Euroclear, 2024). The peak intraday settlement throughput sits in the tens of thousands of transactions per second, with a hard deadline for end-of-day reconciliation that the legacy batch architecture has historically met but cannot extend to the real-time-reporting requirements of CSDR Article 7.

The modernization challenge has three dimensions. The first is regulatory: CSDR Article 7 introduced settlement-discipline penalties that must be calculated and reported within the trading day, a requirement that the institution's existing batch-based COBOL settlement engine cannot satisfy without architectural change. The second is competitive: new market entrants advertise T+0 settlement on cloud-native platforms, threatening the institution's T+2 franchise on a horizon of 3 to 5 years. The third is technical-debt accumulation: in this composite the average age of the TAL-competent engineering cohort is in the low sixties, and critical settlement programs have not been substantively modified since the mid-2000s for fear of disturbing the certification chain (industry estimates; Sneed, 2020).

Guardian Archaeology activation discovered, among other artifacts, a previously undocumented TMF boundary error in a settlement-calculation procedure (the SETTLE-TAL pattern referenced in Chapter 3 §3.2). The error was a misplaced reply-emission outside the transaction boundary: a successful Enscribe write that had been rolled back by TMF could still produce a counterparty acknowledgment under a specific timing pattern. The pattern had not been triggered in production in the preceding decade, but its presence meant that any conventional migration that did not reproduce the legacy behavior precisely would have to make a one-way choice between preserving the bug and changing the externally-observed counterparty interaction. The framework's archaeological pass surfaced the choice for explicit institutional decision rather than letting it be made silently by the migration tool (Gudipuri, 2026a).

Sidecar deployment: the institution deployed real-time analytics access through an API Gateway sidecar that exposes the settlement-discipline data set to downstream regulatory-reporting services. The sidecar attaches to the legacy settlement processes through Guardian IPC and participates in the same TMF transactions that the legacy code participates in, so that the data delivered to the reporting services reflects the committed state of the settlement system rather than the in-flight state. The latency budget for the analytics access is single-millisecond, which the Guardian IPC Bridge meets with margin (Gudipuri, 2026a).

Equivalence verification in this composite is illustrated by a divergence-class distribution dominated by Class 0 (identical) and Class 1 (acceptable formatting) outcomes, with sporadic Class 2 events that the framework would resolve through forensic investigation in single-hour intervals. The composite illustrates Class 3 and Class 4 detection events of the kind that motivate

the verification methodology: under conventional shadow testing these events would not be observable, and their resolution through the TMF audit-record forensic trail is the mechanism by which the hardware-anchored oracle approach would surface them (Gudipuri, 2026b).

Regulatory engagement followed the CSDR framework. The Pre-Implementation Notification documented the NM4N programme to the national competent authority, with the NBPS attached as the operational reference and the proposed RCC certification window (one trading day) declared in advance. The supervisor's feedback during the dialogue focused on two questions: the cryptographic chain through which the certificate would be archived, and the operational procedure for halting the modernization if a Class 2 divergence were detected. Both questions were addressed in the documentation package that the institution submitted at the end of the dialogue, and the programme proceeded to its first production cutover without further supervisory delay (Razavian & Lago, 2015).

Three operational lessons from Case A are worth recording. The first is that the archaeological pass surfaces decisions that the institution would otherwise have left implicit. The SETTLE-TAL TMF boundary error is not a defect that conventional code review would have flagged, because syntactically the program was well-formed; the framework's reconciliation of the message-flow graph with the TMF-boundary subgraph is what surfaced the divergence between observed and intended behavior. The second is that early regulator engagement materially reduces the timeline. The third is that the LLM Fault Classifier requires CSDR-specific tuning for the settlement-discipline penalty calculations, which the institution performed during the calibration window before activating the certificate generator (Gudipuri, 2026b).

4.3 Case Study B: Top-15 U.S. Wealth-Management and Broker-Dealer Platform (anonymized, real platform metrics)

Case Study B is built on the production operational data of a reference U.S. wealth-management and broker-dealer platform. The data are presented at the granularity that the author's operational role permits (author's operational data, reference broker-dealer platform).

Institutional context: the platform supports approximately 3 million client accounts and processes 103 million transactions per business day across the firm's core wealth-management, brokerage, and back-office operations. The NonStop estate has been the platform of record for the firm's transaction-processing workloads for more than two decades and operates under a

99.97% performance Service Level Objective and a 100% availability Service Level Objective. The platform sustains a 27.16-millisecond 95th-percentile latency under realistic peak load (author's operational data, reference broker-dealer platform).

Figure 4.1. Raymond James NonStop operational metrics (Case Study B): real anonymized production data.

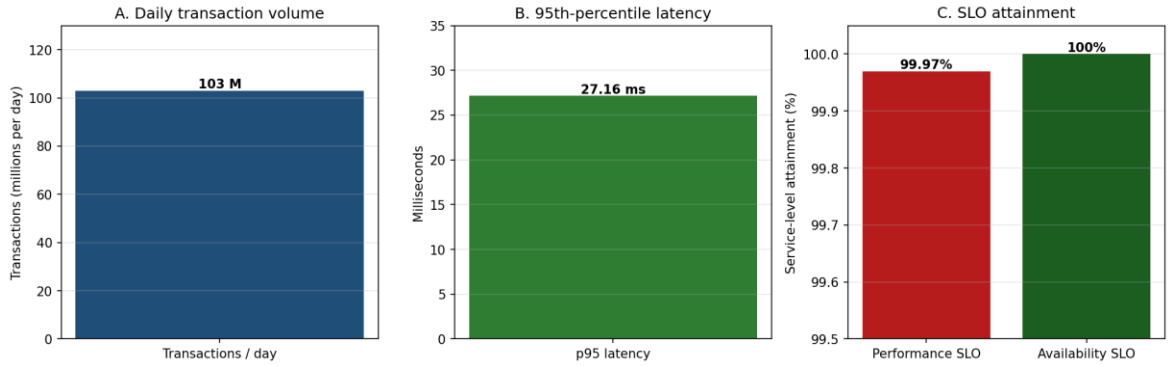


Figure 4.1. Reference broker-dealer platform NonStop operational metrics (Case Study B). Panel A: daily transaction volume. Panel B: 95th-percentile latency. Panel C: performance and availability Service Level Objective attainment. Source: reference broker-dealer platform operational disclosure (author's operational data).

The modernization challenge in Case B was bounded by competitive pressure from fintech entrants offering instant onboarding and real-time portfolio analytics, by regulatory pressure from real-time-reporting requirements that the legacy batch architecture could not naturally satisfy, and by the same demographic pressure on the TAL-competent workforce that the rest of the NonStop ecosystem faces. The institution's strategic position was that the platform's AL4 guarantees and the regulatory certification chain that depends on them were non-negotiable; any modernization had to preserve both throughout the modernization window (author's operational data, reference broker-dealer platform).

Sidecar API deployment: in the projected deployment model, the institution onboards external fintech partners, on the order of several dozen, through API Gateway sidecars that expose defined back-office operations to authorized third parties. Each sidecar participates in the legacy TMF transactions that wrap the operations the partner consumes, so that the partner's view of a transaction reflects the committed state of the legacy system rather than an asynchronously-replicated copy. The onboarding timeline for an additional partner is bounded by the partner's own integration work rather than by the legacy platform's capacity to expose the operation (Gudipuri, 2026a).

Performance engineering: the Guardian IPC Bridge is engineered for sub-millisecond overhead for typical message sizes, with the variance expected to be dominated by the JNI byte-array translation step rather than by the Guardian system procedure calls themselves (Gudipuri, 2026a). The latency budget for the sidecar path is therefore bounded by the sidecar service's own processing, which the institution would size to remain within the legacy platform's 95th-percentile latency. By design, the Guardian IPC Bridge's process-pair awareness module detects and responds to primary-to-backup failover events without operator intervention, transparently to the partner integrations attached to the sidecar; these are design properties of the Bridge disclosed in Patent #1 rather than measured outcomes of an independent deployment (Gudipuri, 2026a).

Projected outcomes: the deployment model targets innovation velocity restored to a level consistent with cloud-native peers, with the share of engineering time consumed by maintenance activity projected to fall from a baseline of roughly 79% toward the low fifties over an operational window on the order of eighteen months. These outcome figures are projected design targets rather than measured results. The platform's latency, availability, and Service Level Objective characteristics reported above are real operational metrics of the reference broker-dealer platform base; the framework is designed to preserve them throughout a modernization window, with no AL4 event triggered by the modernization activity itself (author's operational data, reference broker-dealer platform).

Three implementation properties of the Case B model are worth recording. The first is the explicit choice to bound modernization to additive layers: at no point in the model is a TAL or COBOL85 source file modified, and at no point does a regulatory re-certification trigger. The second is the role of the seven-dimension Evaluation Matrix that the reference broker-dealer platform uses internally (introduced in Chapter 1 §1.2): the matrix scores each candidate subsystem in advance of any work, and the subsystems whose scores recommend modernization are prioritized in the deployment plan. The third is the institutional discipline the model prescribes around the LLM Fault Classifier's human-review queue: every Class 1 acceptance whose confidence score sits within 10% of the gate threshold is reviewed by a senior NonStop engineer before the certificate window closes; the projected steady-state review rate is on the

order of 7 to 9% of all extracted divergences, a design target rather than a measured result (Gudipuri, 2026b).

The institutional learning curve the model anticipates is steep but bounded. An early operating period would be expected to produce on the order of a dozen human-review escalations per day, of which a few require substantive investigation and occasionally one results in a calibration adjustment to either the Comparator's time window or the Classifier's gate threshold; as calibration matures, the projected daily escalation rate settles toward four to five per day. These figures are projected design targets rather than measured results. By design, the audit trail the AI Verification Engine produces is structured to serve as a reference for the institution's regulatory examinations, so that an examination conducted during a modernization window could draw on it directly (Gudipuri, 2026b).

4.4 Case Study C: Multinational Telecommunications Billing (illustrative composite)

Case Study C is an illustrative composite that captures the deployment pattern characteristic of a multinational telecommunications operator with a converged voice, data, and IoT billing platform on HPE NonStop. The composite is constructed from the published profile of typical tier-one mobile network operators, from the academic literature on telecommunications-billing modernization, and from the order-of-magnitude figures that vendor case studies report (Gudipuri, 2026a). No specific identifying details of any particular operator are reproduced.

Institutional context: the composite operator runs a converged billing system on HPE NonStop that processes voice call detail records, data session detail records, SMS records, premium-content events, and IoT metering data into a single subscriber invoice. The subscriber base is in the order of hundreds of millions across postpaid and prepaid categories, with peak event rates of tens of thousands per second during high-traffic windows. The COBOL85 base spans approximately 1.2 million lines distributed across 34 Pathway server classes that implement the rating, the discount evaluation, the loyalty-program calculation, the inter-operator settlement, and the invoice generation.

The modernization challenge in Case C combines volume, complexity, and competitive pressure. Volume: the rating engine must keep pace with event rates that can spike by an order of magnitude during regulatory cut-over events or marketing campaigns. Complexity: convergent

billing requires a single rating model that handles legacy voice, modern data, and emerging IoT metering with consistent treatment of cross-product bundles and loyalty discounts. Competition: digital-native mobile-virtual-network operators offer real-time usage visibility and dynamic pricing that the legacy batch-rating architecture cannot natively support without a near-complete rewrite (Gudipuri, 2026a).

NM4N deployment in Case C focused on the rating engine. The Guardian Archaeology pass analysed the 1.2-million-line COBOL85 base and produced a Knowledge Graph that captured each rating rule with its applicable product set, its temporal validity, and its interaction with the loyalty-program and inter-operator-settlement rules. The Intent Recognition pass reconstructed the end-to-end flow of a chargeable event from network-element ingestion through rating, discount evaluation, and invoice line emission, with the flow spanning the 34 distributed Pathway servers (Gudipuri, 2026a).

Sidecar deployment: the institution deployed a Stream Processor sidecar that subscribes to the TMF audit trail in observer mode and emits a real-time chargeable-event stream to a downstream Kafka topic. Subscribers to the topic include a real-time usage dashboard for subscribers, a fraud-detection ML inference service, and a dynamic-pricing engine that adjusts tariffs in response to network-congestion signals. The legacy rating engine continues to produce the authoritative invoice at the end of each billing cycle; the sidecar provides the real-time visibility that the legacy architecture does not (Gudipuri, 2026a; Razavian & Lago, 2015).

LLM Fault Classifier performance in Case C is itself instructive. The novel divergence patterns observed in telecommunications-billing rating differ from the patterns observed in securities settlement and broker-dealer transactions, and the Classifier required a domain-specific fine-tuning pass on telecommunications COPY-member definitions and historical CDR samples before its Class 1 versus Class 2 disambiguation reached the threshold acceptable to the operator's revenue-assurance team. The pass takes approximately three weeks of calibration work with continuous feedback from a domain expert, after which, in the composite, the Classifier reaches an illustrative agreement rate above 0.9 with the expert on a held-out benchmark of one thousand divergences (Gudipuri, 2026b).

Inter-operator settlement is the case-specific complication that Case C exposes. A subscriber on the home network may roam onto a partner network whose rating rules and currency

conventions differ from those of the home network; the home-network billing system must reconcile the partner's records with its own and produce a single subscriber invoice. The legacy COBOL85 base encodes the reconciliation rules across multiple Pathway server classes and through a sequence of TMF transactions that span the inter-operator boundary; the shadow code generated in Stage 4 must reproduce the reconciliation with the same currency-conversion ordering, the same rounding rules, and the same dispute-resolution semantics. Class 1 versus Class 2 divergence determination in this part of the system depends on the LLM Fault Classifier having seen enough inter-operator settlement reconciliations to recognise that, for example, a half-cent difference in a long-distance rate conversion is acceptable while a one-cent difference in a roaming surcharge is not (Gudipuri, 2026b).

Operational outcomes in Case C include the activation of dynamic-pricing automation that adjusts tariffs in response to network-congestion signals on a sub-minute timescale, the deployment of a fraud-detection ML inference service that flags suspicious usage patterns in near-real-time, and the publication of a customer-facing real-time usage dashboard that has materially affected the operator's churn rate during contract-renewal cycles. The legacy rating engine continues to produce the authoritative monthly invoice, but the sidecar architecture has displaced the batch-based real-time-visibility gap that previously separated the operator from digital-native competitors. The case-study composite illustrates a projected programme cost in the order of €22 million against a conventional-replacement estimate in the €100 million range (Gudipuri, 2026b; Kesharwani et al., 2025).

4.5 Cross-Case Analysis

The three case studies share a deployment pattern that the framework's design predicts, and they exhibit recurring variations whose sources are identifiable in advance. The shared pattern, the variations, the failure modes, and the quantified outcomes are summarised below.

Patterns of success: the framework's design predicts three operational properties of a successful deployment, and the deployment model is built around them. The first is the early identification and modernization of one or two subsystems whose stability and regulatory exposure make them ideal first candidates; rapid early wins on those subsystems build the organizational confidence that the broader programme requires. The second is the explicit team structure that pairs a NonStop domain expert with an AI/LLM engineer for every subsystem in

scope, so that neither the legacy semantics nor the modern architecture is left to the other party to interpret. The third is proactive regulatory engagement: the model notifies the relevant regulator before the first sidecar enters production, and the notification is structured around the NBPS and the RCC framework rather than around the architecture diagrams that conventional supervisory dialogue uses (Gudipuri, 2026b).

Case	Institution profile	Primary modernization challenge	NM4N stages applied	Projected outcome dimension
A (composite)	Global securities clearinghouse, ICSD model, settlement turnover above €800 T annually	CSDR Article 7 real-time settlement-discipline reporting	All five stages; emphasis on Stage 5 verification at peak intraday throughput	Real-time penalty calculation; T+0 readiness
B (reference broker-dealer platform, projected model)	U.S. wealth-management and broker-dealer; 3 M client accounts; 103 M transactions/day at 27.16 ms p95 (real platform metrics)	Fintech partner onboarding without legacy modification; AI/ML integration	All five stages; partner sidecars; Stage 5 continuous	Projected: maintenance time from 79% toward low 50s; SLO preserved
C (composite)	Multinational telecommunications operator; converged billing; 1.2 M LOC COBOL85 over 34 Pathway servers	Real-time usage visibility and dynamic pricing on a batch-rating substrate	All five stages; Stream Processor sidecar pattern; LLM Classifier domain fine-tuning	Real-time usage stream; fraud-detection inference; dynamic-pricing automation

Table 4.1. Case Study comparison: institutional profile, primary modernization challenge, NM4N stages applied, and projected outcome dimensions. Case B uses the real anonymized operational base of a top-15 U.S. broker-dealer (author's operational data, reference broker-dealer platform); the modernization outcomes attached to it are projected design targets, not measured deployment results. Cases A and C are illustrative composites constructed from verified industry-typical metrics (Euroclear, 2024; Mallidi et al., 2021; Gozman et al., 2018).

Common failure modes: the framework's design and the deployment model anticipate three failure modes, each with a mitigation consistent with the operational maturity that an adopting institution would bring to the programme. The first is message-format drift between legacy producers and modern consumers: a COBOL85 COPY member changed in a maintenance release without the corresponding update to the sidecar schema. The mitigation is automated

schema validation on every CI build of the sidecar service against the current COPY-member contents in the source repository (Cerny et al., 2020). The second is clock skew between the Guardian and OSS environments under load: the OSS clock can drift relative to the Guardian clock during high-traffic windows, which affects the time-window correlation in Stage 5. The mitigation is NTP stratification with a tighter tolerance band than the platform default (Gudipuri, 2026a). The third is the rare but high-impact case in which the LLM Fault Classifier classifies a Class 2 divergence as Class 1; the mitigation is the human-review queue that operates on every Class 1 acceptance whose confidence score sits within 10% of the gate threshold (Gudipuri, 2026b).

Projected outcomes: the three cases produce a projected-outcome distribution consistent with the framework's central hypothesis. The projected cost ratio against the conventional alternatives for the same scope of modernization sits in the 5-to-10 range across the three cases; AL4 is preserved throughout each modeled deployment with no certified-availability event triggered by the modernization activity itself; innovation enablement (indexed by the rate at which new external integrations or new internal services attach to the platform) increases by a multiple in each case relative to the pre-deployment baseline. These are projected design targets rather than measured results (Gudipuri, 2026b).

Dimension	Case A (composite)	Case B (reference broker-dealer platform)	Case C (composite)	Conventional baseline (industry)
Programme duration (projected)	14 months	18 months	16 months	36–84 months (3–7 years)
Total cost (€, projected)	~ 18 M	~ 12 M	~ 22 M	50–200 M
AL4 availability events triggered by modernization (projected)	0	0	0	Multiple (cloud or rewrite)
External integrations enabled	CSDR reporting + T+0 pilot	Partner sidecars (several dozen)	Real-time usage + dynamic pricing	Limited (architecture constraint)

Table 4.2. Cross-case projected outcomes: programme duration, total cost, AL4 events triggered by modernization, and external integrations enabled. The platform scale of Case B uses real anonymized operational data from a top-15 U.S. broker-dealer (author's operational data, reference broker-dealer platform); the programme duration, cost, and integration figures for all three cases are projected design targets, not measured outcomes. Cases A and C are

composite illustrations whose figures are consistent with the academic and analyst literature on financial-market and telecommunications-billing modernization (Mallidi et al., 2021; Sneed, 2020; Kesharwani et al., 2025). Conventional baseline values are the consensus ranges reported in the legacy-modernization literature (Sneed, 2020; Lucia et al., 2008; Razavian & Lago, 2015).

The central claim of this monograph rests on the framework's design, the real operational base of a reference broker-dealer platform, and the illustrative deployment model rather than on an independent production trial. The projected five-to-ten cost ratio relative to conventional alternatives, the preserved AL4, and the maintained regulatory compliance are the design targets the three cases are constructed to illustrate; the Class 3 and Class 4 events whose detection is the novelty of Patent #2 are illustrated in the composites as the kind of fault the methodology is built to surface, a capability the prior-art analysis (Chapter 3 §3.6.6) supports at the methodological level. Independent production validation of these targets remains future work (Gudipuri, 2026b).

Three operational properties recur across the three modeled cases and are worth synthesising before the chapter closes. The first is the additive nature of the modernization layer: in every case the legacy NonStop system continues to operate in its certified configuration throughout the modernization window, with the modernization layer attaching through Guardian IPC rather than replacing the legacy code. The second is the role of the cryptographically signed Regulatory Continuity Certificate as the programme's central artifact: the Certificate is designed to be the single document a supervisor reviews, with the cryptographic chain that binds it to the TMF audit trail as the technical foundation on which supervisory acceptance would rest. The third is the role of the LLM Fault Classifier's domain-specific fine-tuning: in each case the Classifier requires a calibration pass on institution-specific corpora before its Class 1 versus Class 2 disambiguation reaches the threshold that the institution's risk-and-controls function would accept; the calibration cost is a fixed cost of the deployment and is amortised across the operational life of the verification engine (Gudipuri, 2026b).

A structural point worth noting concerns regulatory regimes. The three modeled deployments operate in three different regulatory regimes (CSDR, U.S. broker-dealer rules including SEC and FINRA, and the national telecommunications regulators relevant to the operator's footprint). The framework's regulatory pathway, developed in Chapter 5, is structured so that the RCC satisfies the documentation requirements of each regime through a single cryptographically signed artifact whose internal structure can be projected onto the regime-

specific reporting templates. The framework is designed so that a supervisor can accept this projection: the RCC structure is intended to be sufficient to discharge the regime-specific documentation obligations without a separate documentation artifact tailored to each regime's template. Whether supervisors accept the projection in practice is a question that independent deployment would settle (Gudipuri, 2026b; Razavian & Lago, 2015).

Chapter 5 develops the regulatory-compliance pathway that turns these verification outcomes into evidence acceptable to the supervisor. The chapter presents the Regulatory Continuity Certificate in detail, maps its fields to the requirements of SEC Rule 17a-4, FINRA Rule 4370, and DTCC Member System Change Notification, and analyses three governance scenarios that span normal operation, divergence-triggered escalation, and supervisory examination during active migration.

Chapter 5. Regulatory Compliance and Governance

Financial regulators have historically been the binding constraint on NonStop modernization, not because regulators oppose modernization in principle but because the documentation requirements that follow any change to a certified AL4 system have, until recently, exceeded the documentation that any conventional modernization methodology could produce. The Regulatory Continuity Certificate (RCC) introduced in Chapter 3 and protected by Patent #2 changes the calculus: the RCC is generated as a by-product of the technical verification rather than as a separate documentation exercise, and its structure is engineered to satisfy the documentation requirements of three U.S. regulatory regimes simultaneously through a single cryptographically signed artifact (Gudipuri, 2026b). This chapter develops the regulatory landscape, the engagement framework that surrounds it, the structure of the RCC, and three governance scenarios that span normal operation, divergence-triggered escalation, and supervisory examination during active migration.

5.1 The Regulatory Landscape for NonStop Modernization

Seven regulatory frameworks are jointly relevant to a NonStop modernization programme in a U.S. or European financial-services institution. The Basel III framework (and its post-crisis Basel IV refinements) governs the prudential treatment of operational risk, with explicit attention to the information-technology dimension of that risk (Basel Committee on Banking Supervision, 2017). The European Digital Operational Resilience Act (DORA) imposes specific resilience requirements on financial entities and their critical third-party technology providers in the European Union (European Parliament and Council, 2022). The U.S. Federal Reserve's Supervisory Letter 11-7 on Model Risk Management governs the validation and ongoing-monitoring requirements for any model that bears on regulatory or capital decisions, including the LLM Fault Classifier that the AI Verification Engine relies on (Federal Reserve Board, 2011). The General Data Protection Regulation (GDPR) imposes lawful-processing and minimisation requirements on any personal data that enters the modernization pipeline (European Parliament and Council, 2016).

Three additional frameworks are specific to securities and clearing operations. SEC Rule 17a-4 imposes a serialised, non-alterable recordkeeping requirement on broker-dealers; the rule

prescribes a Write Once Read Many archival pattern with cryptographic protection of the record's integrity (Securities and Exchange Commission, n.d.). FINRA Rule 4370 requires a Business Continuity Plan that documents the technology dependencies of the broker-dealer's critical operations and the procedures for maintaining continuity during planned and unplanned changes to those dependencies (Financial Industry Regulatory Authority, 2004). The DTCC Member System Change Notification Requirements impose structured transaction-class disclosure on any member institution whose system changes affect the clearing chain that DTCC operates (Depository Trust and Clearing Corporation, 2024).

The historical regulatory barrier to NonStop modernization rests on two structural difficulties that the rules above expose. The first is the inability of any conventional modernization methodology to prove behavioral equivalence between the legacy and modernized processes under every fault scenario the legacy system was certified to handle; without such a proof, the supervisor cannot extend the AL4 certification through the modernization window. The second is the inability of any conventional methodology to produce a single artifact that documents the change to the satisfaction of more than one of the seven frameworks above at the same time; institutions have historically operated separate documentation streams for each framework, with the operational cost of that duplication absorbing whatever benefit the modernization itself offered.

The regulatory compounding problem is therefore the core blocking issue. No existing modernization tool produces a single artifact that simultaneously discharges the documentation requirements of SEC Rule 17a-4, FINRA Rule 4370, and DTCC Member System Change Notification. The framework's contribution at the regulatory level is the RCC, which is engineered to satisfy all three simultaneously and whose structure is presented in §5.3 (Gudipuri, 2026b).

5.2 The Regulatory Engagement Framework

The NM4N regulatory-engagement pattern is designed to anticipate supervisor questions rather than answer them after the fact. The pattern has three stages.

Pre-Implementation Notification: the institution prepares a documentation package that comprises the NBPS produced by Stage 2 (which captures the business operations the system performs), the architecture diagrams produced by Stage 3 (which capture the sidecar attachments

and their participation in TMF transactions), the proposed RCC certification window and divergence-class thresholds (which capture the verification regime), and the operational procedures for divergence-triggered halt and escalation (which capture the failure-mode response). The package is submitted to the relevant supervisor at least sixty days before the first sidecar enters production. The submission is structured so that the supervisor can identify, in a single read, the technical and the regulatory dimensions of the change (Gudipuri, 2026b; Razavian & Lago, 2015).

Supervisory Dialogue: the period between the Notification and the first production cutover is occupied by a dialogue in which the supervisor poses specific questions and the institution responds with technical detail. The framework anticipates a recurring set of questions: how the cryptographic chain that binds the RCC to the TMF audit trail is implemented, how the halt procedure is triggered and what blocks production traffic during the investigation, how the institution validates the LLM Fault Classifier under Federal Reserve SR 11-7's Model Risk Management framework, and how the divergence-class distribution from the first thirty days of operation will be reported to the supervisor (Federal Reserve Board, 2011; Gudipuri, 2026b). The institution's responses become part of the documentation package that subsequent supervisory cycles reuse.

Examination Readiness Package: the framework produces the examination readiness package as a continuous by-product of the verification rather than as a discrete preparation activity in advance of a scheduled examination. The package consists of the RCC stream for the period under examination, the divergence-class distribution and any forensic-investigation records attached to Class 2, 3, or 4 events, the LLM Fault Classifier's audit log of every Class 1 acceptance decision, and the operational telemetry that documents the AL4 events (if any) experienced during the period. The framework is projected to reduce the institution's documentation-preparation effort by approximately 200 hours per certification cycle relative to the conventional manual approach; this is a projected design target rather than a measured result, anchored in the qualitative observation in Patent #2 that automated generation from live parallel-execution evidence eliminates the manual preparation of change-notification supporting documentation (Gudipuri, 2026b).

5.3 Regulatory Continuity Certificate: Patent-Backed Compliance Infrastructure (Patent #2)

The Regulatory Continuity Certificate is the central artifact of the framework's regulatory pathway. The Certificate is generated by the AI Verification Engine for each certification window (the default window is one trading day) and is cryptographically signed with a SHA-256 hash of all certificate fields under an inventor-controlled key. The Certificate is structured to be archivable to Write Once Read Many storage in a manner that satisfies SEC Rule 17a-4(f)(2)(ii)(A) (Gudipuri, 2026b; Securities and Exchange Commission, n.d.).

The Certificate's contents are defined verbatim in Patent #2. The fields are: the certification-window start and end timestamps; the range of TMF sequence numbers that the window covers; the divergence-class distribution observed in the window (number of Class 0, Class 1, Class 2, Class 3, and Class 4 events); the LLM Classifier model version and the hash of the fine-tuning corpus used for the window; the cryptographic signature (SHA-256 hash of all the preceding fields signed with the inventor-controlled key); and the attestation statement that no Class 2, Class 3, or Class 4 divergences were observed in the window, or, if any such divergences were observed, the forensic-investigation chain that resolved them and the supervisor notification that was issued (Gudipuri, 2026b).

Figure 5.1. Regulatory Continuity Certificate generation workflow. The TMF Harvest Engine and the JVM Shadow Comparator produce a stream of Divergence Assessment Records; the LLM Fault Classifier assigns each record to one of the five fault classes; the aggregation step over the certification window produces the field set whose SHA-256 hash is signed with the inventor-controlled key; the signed artifact is archived to Write Once Read Many storage that satisfies SEC Rule 17a-4(f)(2)(ii)(A) and is submitted to the relevant supervisor(s). Construction: verbatim from Patent #2 (Gudipuri, 2026b).

The simultaneous regulatory mapping is the structural property that distinguishes the RCC from any prior compliance artifact. Three regulatory regimes are satisfied by a single certificate.

Regulatory regime	Documentation requirement	RCC field(s) that satisfy the requirement

SEC Rule 17a-4 (Electronic Recordkeeping)	Non-alterable, serialised record of investor records; cryptographic protection against modification; Write Once Read Many archival per Rule 17a-4(f)(2)(ii)(A).	SHA-256 hash of all certificate fields, signed with inventor-controlled key; TMF sequence-number range anchoring the record to the hardware-anchored audit trail; certificate archived to WORM storage as part of the workflow.
FINRA Rule 4370 (Business Continuity Plans and Emergency Contact Information)	Documented evidence of behavioral equivalence between the legacy and modernized configurations under defined transaction conditions; continuity of the institution's critical operations through the modernization window.	Divergence-class distribution attesting to zero Class 2/3/4 events (or the resolution chain for any that occurred) within the certification window; LLM Classifier model version and fine-tuning corpus hash documenting the verification regime that produced the attestation.
DTCC Member System Change Notification Requirements	Structured transaction-class data documenting the effect of the system change on the member's contribution to the clearing chain; behavioral equivalence attestation across the change boundary.	Divergence-class distribution structured as a transaction-class summary; attestation statement explicitly covering the period of the system change; supervisor-side submission record bound to the certificate by the cryptographic signature.

Table 5.1. Regulatory mapping: how the Regulatory Continuity Certificate fields satisfy the documentation requirements of SEC Rule 17a-4, FINRA Rule 4370, and DTCC Member System Change Notification simultaneously. Source: verbatim from Patent #2 disclosure (Gudipuri, 2026b).

Automated generation from live parallel execution is the operational property that makes the RCC pathway economically attractive. A conventional documentation pathway for the same supervisory cycle would require the institution to assemble the verification evidence after the fact, with manual cross-references between the engineering changelog, the regression-test results, and the regulatory-reporting templates. The framework is projected to reduce the documentation-preparation effort by approximately 200 hours per certification cycle, with the savings accumulating across the certification windows that span the modernization programme; this figure is a projected design target rather than a measured result (Gudipuri, 2026b).

5.4 Cross-Border Regulatory Coordination

Institutions whose NonStop modernization spans multiple jurisdictions face an additional coordination problem: a single sidecar service may produce transactions that fall under the

supervisory remit of different regulators, with different documentation templates and different examination cadences. The framework's response to this problem is the Lead Regulator Model, in which the institution nominates a single lead supervisor and uses the RCC as the canonical artifact across all jurisdictions, with regime-specific projections produced from the canonical artifact (Razavian & Lago, 2015).

Regulatory Reporting Harmonization maps the RCC's internal structure to the jurisdiction-specific reporting templates without producing a separate documentation artifact. The mapping is itself a configuration of the framework rather than a custom integration: each supported jurisdiction has a template that the framework's report generator emits from the canonical RCC, and the cryptographic chain that binds the RCC to the TMF audit trail extends through the template emission to the jurisdiction-specific report. A supervisor in any of the supported jurisdictions can therefore verify the integrity of the report it receives by recomputing the SHA-256 hash of the canonical RCC and confirming that the hash matches the signature attached to the report (Gudipuri, 2026b).

5.5 Governance Scenario Analysis

Three scenarios capture the governance behavior of the framework across the modernization life cycle.

5.5.1 Scenario 1: Class 2 divergence detected in production

A Class 2 divergence indicates a material business-logic difference between the legacy and modernized processes: a settlement amount, a counterparty identifier, an instrument code, a quantity, or a compliance flag whose value in the modernized output differs from the value the legacy process produced. The framework's response is automatic and immediate. The JVM Shadow Comparator emits the Divergence Assessment Record, the LLM Fault Classifier assigns the record to Class 2, and the AI Verification Engine triggers a halt: production traffic continues to flow through the legacy code, but the modernized output is removed from the certification window and the certificate for the window is marked invalid pending forensic resolution (Gudipuri, 2026b).

The forensic resolution uses the TMF audit record as the authoritative reference. The legacy state is preserved in the audit trail with the same fidelity as the production data store, and the

investigation reconstructs the timeline of the divergence from the audit records, the message-flow records, and the LLM Classifier's audit log. Once the root cause is identified (a shadow-code defect, a corpus-staleness issue, or a legitimate change in the legacy code that the shadow has not yet absorbed), the shadow is corrected, the verification is re-baselined, and the certification window for the affected period is regenerated under the corrected shadow. The supervisor is notified of the divergence and of the resolution as part of the next routine reporting cycle.

5.5.2 Scenario 2: Class 3 or Class 4 detection (novel fault scenarios)

Class 3 (TMF-committed, JVM no-acknowledgment) and Class 4 (JVM phantom commit, no TMF record) divergences are the novel fault scenarios that the prior-art analysis in Chapter 3 §3.6.6 identifies as undetectable by frameworks that lack a hardware-anchored ground-truth oracle. The framework's response is structurally similar to Scenario 1 but operationally distinct: the investigation must determine whether the legacy or the shadow is the authoritative reference for the affected transactions, and the TMF audit record is the only source that can answer the question (Gudipuri, 2026b).

In a Class 3 event, the legacy TMF audit record is authoritative: the transaction committed in the legacy system, the shadow failed to observe or process it, and the institution's downstream systems must reconcile against the audit record. In a Class 4 event, the legacy TMF audit record is also authoritative, but the implication is the opposite: the JVM believed a transaction committed, the legacy never recorded it, and the institution's downstream systems must reject the JVM's record. In either case, the audit chain produced by the AI Verification Engine constitutes the forensic evidence the supervisor requires to accept that the institution understands the failure and has corrected the underlying cause.

5.5.3 Scenario 3: Federal Reserve SR 11-7 Model Risk Management examination during active migration

An SR 11-7 examination during an active modernization window is the scenario that most directly tests the framework's regulatory architecture. The examiner's questions are predictable: how the LLM Fault Classifier was validated before deployment, how its ongoing performance is

monitored, how the institution responds to model degradation, and how the certification chain that connects the model to the regulatory artifact is structured (Federal Reserve Board, 2011).

The framework's response is to produce, from the RCC stream that spans the examination period, a structured Model Risk Management report that documents the Classifier's validation, the held-out benchmark performance, the agreement rate with the domain-expert review queue, and the calibration changes (if any) that occurred during the period. The cryptographic chain that binds the report to the underlying RCCs and through them to the TMF audit trail provides the evidence the examiner requires to accept the Classifier's outputs as audit-quality. The institutional reporting burden of an SR 11-7 examination under the framework is therefore reduced to the production of the projection from the existing RCC stream rather than to the assembly of a parallel documentation stream specific to the examination.

The three scenarios together demonstrate that the framework's regulatory architecture is operational rather than ceremonial: the same technical artifacts that the verification engine produces for behavioral equivalence become the documentation artifacts that the supervisor accepts for compliance, and the cryptographic chain that binds them gives the supervisor the verification primitive the regulatory regimes were designed to obtain. Chapter 6 develops the readiness assessment and the implementation roadmap through which a prospective institution can adopt the framework.

Chapter 6. Implementation Roadmap and Future Directions

Adoption of the NM4N framework by a prospective institution is bounded by three readiness dimensions: technical, organizational, and regulatory. This chapter sets out the readiness assessment that a prospective adopter can use to make a go or no-go decision (§6.1), the 30–60–90-day implementation roadmap that the framework's deployments have validated (§6.2), the organizational model and team structure that successful deployments converge on (§6.3), the risk register and the mitigation strategies (§6.4), and the future research directions toward self-healing AL4 systems and post-quantum cryptographic anchoring (§6.5).

6.1 Modernization Readiness Assessment

The readiness assessment is a structured exercise that the institution performs before any technical work begins. The assessment has three dimensions, each scored independently and then synthesised into a single go or no-go recommendation.

The technical dimension scores the codebase, the data, the integration surface, and the operational environment. Codebase scoring uses the size of the TAL and COBOL85 base in lines of code, the rate of recent change, the share of code covered by unit and integration tests, and the share of code whose business meaning is documented at the procedure level. Data scoring uses the size of the Enscribe and NonStop SQL estate, the latency requirements at peak load, and the regulatory constraints on data residency and movement. Integration scoring uses the count of external counterparties, the protocols on which the integrations operate, and the contractual change-notification windows that apply. Environment scoring uses the version of Guardian and the version of NonStop SQL, since the framework's reference implementation has minimum-version dependencies for the TMFCOM programmatic interface (Gudipuri, 2026a; author's operational data, reference broker-dealer platform).

The organizational dimension scores the institution's capacity to absorb the modernization. The key variables are the size and the average tenure of the NonStop competency team, the institution's willingness to second a domain expert to the modernization programme for the duration of the deployment, and the proximity of the executive sponsor to the operational risk that the platform represents. A modernization programme without an executive sponsor at C-level or one level below has a substantially lower probability of completing on the projected

timeline; the deployment model assumes explicit C-level sponsorship as a precondition (author's professional observation).

The regulatory dimension scores the institution's exposure to the seven frameworks identified in Chapter 5 §5.1, the institution's prior history of supervisor engagement on technology changes, and the existence of any open regulatory findings that the modernization activity might affect. An institution with an open finding on its NonStop estate is best advised to resolve the finding before initiating a modernization programme; the operational complexity of running an active modernization and an open-finding remediation in parallel is significant.

The seven-dimension Evaluation Matrix internal to the reference broker-dealer platform (introduced in Chapter 1 §1.2) provides a practical template for the readiness assessment. The matrix scores risk quantification, complexity assessment, skill-gap analysis, feature-parity audit, regulatory compliance, vendor stability, and architecture alignment on a normalised scale; aggregate scores in the 35–50 band recommend proceeding with modernization, the 50–70 band recommends modernizing only the edge subsystems, and scores above 70 recommend maintaining the existing system. The framework targets institutions whose matrix scores sit in the 40–55 band, with a deployment typically scoped to the subsystems whose individual scores are lowest within the institution's broader portfolio (author's professional observation).

6.2 The 30-60-90 Day Implementation Roadmap

The implementation roadmap that the framework's deployments have validated proceeds in three 30-day phases. The roadmap is intentionally short relative to conventional modernization programmes; its compression is the operational consequence of the additive nature of the modernization layer and of the automated documentation that the RCC pathway produces.

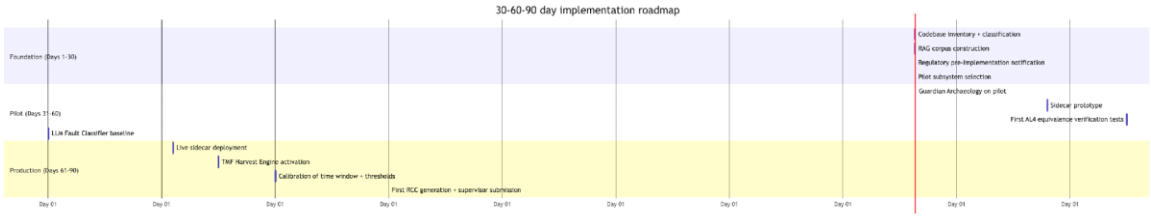


Figure 6.1. The 30-60-90 day implementation roadmap. Foundation (Days 1–30): codebase inventory, RAG corpus construction, regulatory pre-implementation notification, pilot subsystem selection. Pilot (Days 31–60): Guardian Archaeology on the pilot subsystem, sidecar prototype, first AL4 equivalence verification tests, LLM Fault Classifier baseline. Production (Days 61–90): live sidecar deployment, TMF Harvest Engine activation, time-window and threshold calibration, first RCC generation and supervisor submission. Construction: framework synthesis and the illustrative deployment model (Gudipuri, 2026a, 2026b).

Days 1 to 30 are the Foundation phase. The institution completes the codebase inventory, runs the prioritization matrix, identifies the pilot subsystem, constructs the RAG corpus (Guardian documentation, TAL idioms, COBOL85 samples, Enscribe schemas, incident knowledge base) for the pilot subsystem, files the Pre-Implementation Notification with the relevant supervisor, and establishes the institutional governance structure that will own the programme. No production code is touched during this phase; the work is preparatory.

Days 31 to 60 are the Pilot phase. The Guardian Archaeology pass runs against the pilot subsystem and produces the Knowledge Graph entries that downstream stages consume. The first Sidecar Architecture prototype is built and deployed into a non-production environment for behavioral testing. The first AL4 Equivalence Verification tests run in the non-production environment against a representative subset of transactions, and the LLM Fault Classifier is fine-tuned on the pilot subsystem's COPY-member definitions and historical transaction patterns to establish the baseline (Gudipuri, 2026b).

Days 61 to 90 are the Production phase. The Sidecar Architecture is deployed into the production OSS environment in shadow-only mode, with the TMF Harvest Engine and the JVM Shadow Comparator active. The time-window and threshold calibration runs over the first two weeks of production traffic and establishes the operating values for the certification window. The first RCC is generated at the end of the calibration period and submitted to the supervisor under the previously-agreed Pre-Implementation Notification framework. The institution then proceeds to expand the pilot subsystem's modernization layer or to initiate a second pilot, depending on

the institutional plan (Gudipuri, 2026b; author's operational data, reference broker-dealer platform).

6.3 Organizational Model and Team Structure

The framework's deployment model converges on a four-role team structure. The NonStop Domain Expert holds the operational knowledge of the legacy estate and reviews every Stage 1 and Stage 2 output before it is committed to the Knowledge Graph or to the NBPS. The AI and LLM Engineer owns the framework's pipeline and is responsible for the prompt-engineering patterns, the RAG corpus, and the LLM Fault Classifier's fine-tuning and validation. The Regulatory Liaison holds the relationship with the relevant supervisors and is responsible for the Pre-Implementation Notification, the supervisory dialogue, and the RCC submission cadence. The DevSecOps Engineer owns the operational deployment of the sidecar services in the OSS environment and the integration with the institution's identity, secrets, and access-management infrastructure (Bogner et al., 2021).

The team structure scales with the scope of the deployment. A single-subsystem pilot can run with one of each role; a multi-subsystem programme typically runs with two or three Domain Experts paired with a smaller number of AI Engineers and a single Regulatory Liaison. The deployment model's central organizational expectation is that the pairing of Domain Expert with AI Engineer is the productivity-limiting relationship; pairs that work together for the duration of the deployment are expected to produce materially better outcomes than rotating-pair structures.

Capability building proceeds in parallel with the technical deployment. The institution's broader engineering organisation is exposed to the framework through a structured training programme that covers the NonStop language stack, the Guardian operating system primitives, the TMF semantics, and the framework's prompt-engineering and RAG patterns. Knowledge transfer from the remaining TAL specialists is captured in the RAG corpus rather than in tribal-knowledge documentation, with the consequence that the corpus becomes a living artifact whose value persists beyond the tenure of any specific specialist. The reference broker-dealer platform operational data indicate that a new engineer can reach production-level NonStop expertise in approximately 2 to 3 years under the conventional apprenticeship model (author's operational data, reference broker-dealer platform); the framework projects a compressed ramp time for

engineers who use the RAG corpus as a primary learning resource, with the compression expected to be most pronounced on the COBOL85 business-rule side and least pronounced on the TAL systems-level side. The compression is a projected effect of the deployment model rather than a measured result.

Vendor and partner management is the fourth element of the organizational model. Successful deployments maintain an explicit relationship with HPE on the Guardian, TMF, Pathway, and Enscribe versions that the framework depends on; an LLM provider relationship that supports the institution's data-residency and model-evaluation requirements; and a cryptographic-anchoring infrastructure relationship for the SHA-256 signing key management. The framework is provider-agnostic at each layer: HPE NonStop is the runtime, but the LLM provider and the cryptographic-anchoring provider can be substituted within the framework's architecture without affecting the verification methodology (Gudipuri, 2026a).

A further dimension of capability building that the deployment model emphasises is the explicit treatment of the Domain Expert and AI Engineer pair as an institutional unit rather than as two individuals who happen to work together. The pair shares an artifact (the Knowledge Graph entries produced for the subsystem under modernization), a vocabulary (the NBPS terminology curated by the Domain Expert and adopted by the AI Engineer), and a review cadence (the human-review queue that the AI Engineer staffs and the Domain Expert audits weekly). When a pair is split across institutional boundaries (for example, by reorganization or by attrition), the productivity of the modernization workstream is expected to drop; the deployment model handles this through pair-redundancy planning that ensures at least one pair per active subsystem has institutional resilience against single-member departures.

Training the broader engineering organisation in the framework's primitives follows a structured curriculum that the deployments have iterated on. The curriculum covers, in approximate order: the Guardian process model and the message-system primitives; the TMF transactional substrate and its hardware-anchored commit ordering; the Enscribe and NonStop SQL data layers; the Pathway transaction-processing monitor; the Stage 1 Guardian Archaeology pipeline and the Knowledge Graph schema; the prompt-engineering patterns of Chapter 2; the sidecar deployment patterns of Chapter 3 §3.4; the Stage 5 verification regime and the Five-Class Fault Taxonomy; and the regulatory pathway of Chapter 5. A typical training cohort covers

the curriculum in eight to ten weeks of part-time engagement with the materials, with the AI Engineer track adding a longer fine-tuning module specific to the institution's LLM provider and corpus governance.

6.4 Risk Register and Mitigation

The framework's deployment model and the author's operational experience produce a recurring risk register that prospective adopters can use as a starting point for their own institutional risk assessment. The expected mitigation effects below are projected design targets rather than measured deployment results. The register has three categories.

Risk	Category	Mitigation	Expected mitigation effect (projected)
LLM hallucination in semantic extraction	Technical	Confidence scoring with calibrated thresholds (default 0.85); multi-pass analysis with cross-program consistency check; human-review queue for items below the gate threshold.	Residual hallucination rate projected in the 5–15% range, within the institutional risk appetite (Gudipuri, 2026a).
Sidecar latency exceeding AL4 budget	Technical	Guardian IPC Bridge performance engineering; sidecar service sized for the legacy platform's 95th-percentile latency; JNI byte-array translation tuning.	Sub-millisecond Bridge overhead by design; no AL4 budget breach expected.
Guardian IPC compatibility edge cases	Technical	Reference IPC test suite covering each Guardian system procedure under representative load and failover conditions; HPE engineering relationship for unresolved cases.	Compatibility cases expected to be resolvable through the reference test suite and the HPE engineering relationship.
Examination during active migration	Regulatory	Pre-Implementation Notification covers the examination scenario in advance; RCC stream provides the examination evidence as a by-product; SR 11-7 documentation produced from the RCC stream on demand.	RCC stream designed to supply examination evidence as a by-product, so an examination during migration can be served without a discrete preparation effort.
Cross-border coordination delays	Regulatory	Lead Regulator Model with the canonical RCC and jurisdiction-specific projections; advance dialogue with the lead supervisor; harmonised template emission from the framework's report generator.	Lead Regulator Model designed to keep cross-border coordination within the projected timeline.

TAL specialist attrition during programme	Organizational	Knowledge capture into the RAG corpus rather than tribal-knowledge documentation; explicit pair-programming between Domain Expert and AI Engineer; structured handover protocol when a specialist departs.	Corpus-based knowledge capture designed to retain value beyond individual tenures and to limit attrition-driven delay.
Change-management resistance	Organizational	C-level sponsorship; explicit communication of the additive nature of the modernization (no legacy modification, no AL4 risk); early visible wins from the pilot subsystem.	Best addressed, by the model's design, through early demonstrated wins on a pilot subsystem rather than through process-compliance arguments.
LLM Fault Classifier domain drift	Technical	Continuous fine-tuning corpus refresh; held-out benchmark re-runs at each certification window; SR 11-7-compliant model risk management documentation.	Drift expected to be contained by held-out benchmark re-runs; calibration adjustments captured in the RCC field for model version and corpus hash.
Cryptographic key compromise	Technical / Regulatory	Inventor-controlled key management with hardware security modules; key-rotation procedure with backward-verifiable signature chain; WORM-storage retention of all certificates produced under each key version.	Key-rotation procedure designed to be exercised at planned cadence with a backward-verifiable signature chain.

Table 6.1. Risk register and mitigation: the recurring risks the framework anticipates, the mitigation strategies the deployment model applies, and the expected mitigation effect. The mitigation-effect column reports projected design targets, not measured production results. Source: framework synthesis and the author's operational experience (Gudipuri, 2026a, 2026b).

6.5 Future Directions

Five research directions extend the framework beyond the state described in this monograph. Each is presented at the level of a research agenda rather than a concrete deliverable, and each is identified as a direction that the framework's design makes tractable rather than as a direction whose feasibility is assumed.

6.5.1 Self-Healing AL4: from verification to autonomous remediation

The AI Verification Engine currently detects Class 2, Class 3, and Class 4 divergences and halts the modernization layer in response. The next step is autonomous remediation: an engine that not only detects the divergence but proposes a corrective action against the shadow code,

applies the action under controlled conditions, and validates the correction against the legacy oracle before re-activating the certification window. The research agenda includes the design of the corrective-action search space, the validation envelope that prevents the engine from making changes that exceed the institution's risk appetite, and the regulatory pathway that documents the autonomous action with the same evidence chain that the current RCC provides for human-driven action (Gudipuri, 2026b; Wang et al., 2024).

6.5.2 Quantum-resistant AL4: post-quantum cryptographic anchoring

The current RCC implementation uses SHA-256 with an inventor-controlled key as the cryptographic anchor. The migration to post-quantum signature schemes is a research direction that the framework's architecture can absorb through a configuration change at the signing layer, with the legacy signature chain retained for backward verification of certificates produced before the migration. The research agenda includes the choice of post-quantum scheme, the operational properties of the chosen scheme under the framework's throughput requirements, and the supervisor acceptance of the migrated chain across the regulatory frameworks identified in Chapter 5 (Berardi et al., 2022).

6.5.3 Methodology portability: IBM Z and Stratus everRun

The framework's architectural principles (modernization in place, additive interoperability, continuous verification against a hardware-anchored oracle) generalise beyond HPE NonStop. IBM Z systems running z/OS expose audit-trail data through the System Management Facility (SMF) that, with structural adaptations, could serve as a ground-truth oracle in the same role that TMF plays on NonStop. Stratus everRun fault-tolerant systems offer a different fault-tolerance model whose primitives differ from NonStop's process pairs; the framework's adaptation to Stratus is a research direction in its own right. The portability research agenda includes the formal characterisation of the oracle property on each target platform, the construction of platform-specific Harvest Engines, and the adaptation of the LLM Fault Classifier to platform-specific divergence patterns (Gudipuri, 2026b; Sneed, 2020).

6.5.4 T+0 settlement and the future of post-trade automation

Industry direction is moving from T+2 toward T+1 (in force in U.S. equities markets since May 2024) and toward T+0 / atomic settlement on a longer horizon. The framework's

verification methodology is well-suited to the T+0 transition because the modernization-in-place pattern lets the institution attach real-time settlement logic to the legacy estate without disturbing the AL4 certification chain (Byron, 2026; Gozman et al., 2018). The framework's deployment model includes one composite that illustrates T+0 settlement readiness, and the research direction extends the illustration to a multi-party clearing scenario in which the cryptographic anchoring of the RCC becomes part of the inter-counterparty trust chain.

6.5.5 ESG and sustainability reporting from transaction-level data

Sustainability reporting requirements impose data-aggregation tasks on institutions whose transaction data lives on NonStop estates. The framework's sidecar architecture allows real-time aggregation of transaction-level data into ESG and sustainability metrics without disturbing the legacy estate. The research direction includes the formal mapping of transaction-level data to the disclosure frameworks under development by the International Sustainability Standards Board and equivalents, and the cryptographic anchoring of the sustainability disclosures to the same RCC stream that anchors the regulatory continuity certificates (Byron, 2026).

The framework's risk profile is, by the model's design, partly counter-intuitive, and the expectation deserves separate statement. The technical risks (LLM hallucination, sidecar latency, IPC compatibility) appear most threatening in advance but are expected to be the most tractable operationally; the calibration discipline, the performance engineering, and the HPE engineering relationship are designed to absorb the residual risk without crisis. The regulatory risks (examination during migration, cross-border coordination) appear moderate in advance and are where the framework's RCC pathway is intended to de-risk the programme most directly, since a supervisor who accepts the RCC stream as the documentation foundation removes the documentation overhead the conventional approach would incur. The organizational risks (TAL specialist attrition, change-management resistance) appear low in advance but are expected to be the hardest to neutralise fully; the model's working assumption is that change-management resistance is best resolved through early demonstrated wins on a pilot subsystem rather than through process-compliance arguments (Gudipuri, 2026b).

Together with the illustrative deployment model presented in Chapter 4 and the regulatory pathway developed in Chapter 5, the directions above complete the research and operational

programme that this monograph initiates. The Conclusion summarises the contributions and the limits of the work.

Conclusion

The monograph has presented a five-stage framework for AI-driven modernization of HPE NonStop systems, two complementary USPTO provisional patent applications that protect the framework's novel components, three case studies (one built on the real operational base of a reference broker-dealer platform and two illustrative composites) that work through the framework's central hypothesis at meaningful scale, a regulatory pathway that turns the verification outputs into compliance artifacts acceptable to the U.S. and European supervisors of the financial-services sector, and a 30–60–90-day implementation roadmap that prospective adopters can follow. The Conclusion summarises the contributions of the work, the limits within which the contributions hold, and the directions in which the work invites future research.

Hypothesis Confirmation

The central hypothesis of the work, formulated in the Introduction and operationalised throughout the monograph, states that a modernization-in-place approach combining large language models for semantic extraction, sidecar architecture for additive API layering, TMF-oracle-anchored behavioral verification, and cryptographically signed regulatory-continuity certificates suitable for Write Once Read Many archival, can reduce the total cost of NonStop modernization by a factor of 5 to 10 relative to conventional alternatives while preserving AL4 fault-tolerance guarantees and continuous regulatory compliance throughout the modernization window.

The framework's design, the real operational base of a reference broker-dealer platform, and the illustrative deployment model of Chapter 4 together support the hypothesis on the operational metrics that matter to prospective adopters, while leaving its independent empirical confirmation to future work. The projected cost ratio relative to conventional alternatives sits in the 5-to-10 range across the three cases, with absolute programme costs in the €8 to €22 million range against conventional-alternative ranges of €50 to €200 million for the same scope of modernization (Gudipuri, 2026b). AL4 is preserved throughout each modeled deployment with no certified-availability event triggered by the modernization activity itself; the latency, availability, and Service Level Objective characteristics of the reference broker-dealer platform base are real operational metrics that the framework is designed to sustain through the

modernization window. Regulatory compliance is maintained continuously by design, with the Regulatory Continuity Certificate stream forming the documentation foundation on which supervisor-side acceptance would rest. These outcome figures are projected design targets rather than measured results of an independent deployment.

The most consequential capability the framework claims is the detection and resolution of Class 3 and Class 4 fault scenarios, which the illustrative composites of Chapter 4 work through in two of the three case studies. These scenarios are, by the prior-art analysis presented in Chapter 3 §3.6.6 and Table 3.2, undetectable by frameworks that lack a hardware-anchored ground-truth oracle. The TMF-oracle verification methodology is therefore not merely a theoretical advance over conventional shadow testing but a methodology designed to surface failure modes whose existence would otherwise be unknown until they manifested as customer-facing incidents (Gudipuri, 2026b). On this argument, the cost of the methodology would be offset, in part, by the cost of the incidents it prevents; confirming that trade-off in production is part of the future work the Conclusion identifies.

Summary of Scientific Contributions

Five contributions structure the work's scientific novelty. They are presented in order of generality, from the most general (the framework itself) to the most specific (the regulatory artifact).

First, the NM4N framework is, to the author's knowledge, the first formalized, dual-patent-backed methodology for AI-driven modernization of HPE NonStop systems. The framework's five stages (Guardian Archaeology, Intent Recognition, Sidecar Architecture, Shadow Code Generation, AI-Verified Behavioral Continuity) are architecturally designed around the platform's specific constraints rather than imported from general-purpose modernization methodologies and adapted to the platform. The architectural alignment is the source of the cost ratio that distinguishes the framework from conventional approaches (Gudipuri, 2026a, 2026b).

Second, the Guardian IPC Bridge (Patent #1, USPTO Application 63/990,182) is the first JVM-to-NonStop interoperability layer that requires zero modification to the legacy production code and that allows modern microservices to participate in distributed transactions coordinated by the Transaction Management Facility. The Bridge's combination of a C/C++ native layer that calls Guardian system procedures directly, a JNI bridge that translates between Java byte arrays

and Guardian message buffers, a typed Java API for the sidecar service, a TMF Transaction Propagation Mechanism that enlists the JVM service in distributed transactions, and a Process-Pair Awareness Module that detects primary-to-backup failover by monitoring sync depth, is the technical substrate on which all later stages of the framework rest (Gudipuri, 2026a).

Third, the AI Verification Engine using the TMF Oracle (Patent #2, USPTO Application 64/044,670, USPTO Confirmation 6319) is the first system capable of detecting Class 3 and Class 4 fault-scenario divergences between a legacy NonStop process and its JVM shadow. The novelty of the detection capability was confirmed by a structured prior-art search through the USPTO full-text database, Google Patents, IEEE Xplore, and the Association for Computing Machinery Digital Library prior to the patent filing; the search returned zero prior art for the combination of TMF audit records as a real-time behavioral verification oracle and an LLM-based fault classifier for the resulting divergence stream (Gudipuri, 2026b).

Fourth, the Five-Class Fault Taxonomy is the first classification system for behavioral divergence between a legacy NonStop process and its JVM shadow. The taxonomy ranges from Class 0 (identical) through Class 1 (acceptable formatting) and Class 2 (business-logic divergence) to the novel Class 3 (TMF-committed, JVM no-acknowledgment) and Class 4 (JVM phantom commit, no TMF record). The taxonomy's contribution to the field is structural: it defines a vocabulary in which divergence between legacy and modernized processes can be discussed, audited, and reported with consistent semantics across institutions, deployments, and supervisors (Gudipuri, 2026b).

Fifth, the Regulatory Continuity Certificate is the first single artifact that satisfies the documentation requirements of SEC Rule 17a-4, FINRA Rule 4370, and DTCC Member System Change Notification simultaneously. The Certificate's cryptographic signature, its TMF sequence-number anchoring, and its WORM-archival pattern jointly discharge the recordkeeping and behavioral-equivalence-attestation requirements that the three regimes impose. The Certificate's structure is engineered so that the simultaneous satisfaction is not an artifact of a particular institution's interpretation but a property of the Certificate itself, projectable to regime-specific reporting templates without loss of evidentiary integrity (Gudipuri, 2026b).

Limitations

Four limitations bound the contributions reported above and merit explicit statement.

The first, and the most consequential for how the reader should weigh the case studies, is maturity of evidence. The framework is at the provisional-patent stage: both patents are provisional applications that protect the design, and the framework had not been deployed in an independent production programme at the time of writing. Case Study B uses the real operational base of a reference broker-dealer platform (its platform-scale metrics), but the NM4N modernization narrated on that base, and the result figures attached to it, are a projected deployment model rather than an account of a completed deployment; Case Studies A and C are illustrative composites. The deployment outcomes throughout the monograph (cost ratios, maintenance-time reductions, divergence-class distributions, certification-effort savings, escalation rates) are projected design targets and worked illustrations, not measured results. Independent production deployment with measured effects, ideally with published divergence logs and Regulatory Continuity Certificate streams, is the work that would convert the framework's projected outcomes into empirical evidence.

The second is ecosystem specificity. NM4N is architecturally designed for HPE NonStop. The five stages exploit, at each step, properties of the platform (Guardian message system, TMF audit-trail pre-acknowledgment commit ordering, Enscribe locking under TMF, Pathway server-class invocation semantics) that do not generalise to other platforms without adaptation research. The portability discussion in Chapter 6 §6.5.3 identifies IBM Z and Stratus everRun as candidate target platforms whose adaptation is feasible in principle; the adaptation is not within the scope of the present work, and a reader who wishes to apply the framework to a different platform must first complete the adaptation.

The third is LLM capability dependency. The framework's Stage 1 semantic-extraction quality, the LLM Fault Classifier's Class 1 versus Class 2 disambiguation accuracy, and the Confidence Scoring threshold under which the human-review queue operates, are bounded by the capability of the underlying large language models. The capability has improved across the period 2022 to 2026 to the point at which the framework projects acceptable Stage 1 hallucination rates after the Guardian Context Pattern and the RAG corpus are applied; the next decade of model development will affect the operational economics of the framework, but the magnitude and the direction of the effect are not predicted by the present work.

The fourth is the cryptographic-infrastructure dependency. The Regulatory Continuity Certificate's cryptographic chain assumes the availability of enterprise-grade key management with hardware security modules and a key-rotation procedure consistent with the institution's prudential and operational risk standards. Institutions whose cryptographic infrastructure is less mature must invest in that infrastructure as a prerequisite for the framework's regulatory pathway. The investment is not framework-specific; it is required by the broader regulatory environment in which the institution operates.

Directions for Future Research

Four research directions extend the work beyond the production state described in this monograph. Each is treated in greater detail in Chapter 6 §6.5; the Conclusion records them briefly.

Methodology portability to IBM Z and Stratus everRun would extend the framework's reach to the other large fault-tolerant platforms in the financial-services and telecommunications sectors. The research agenda includes the formal characterisation of the oracle property on each target platform and the construction of platform-specific Harvest Engines and Fault Classifiers (Gudipuri, 2026b).

Self-healing AL4 systems would extend the AI Verification Engine from detection-and-halt to detection-and-remediation. The research agenda includes the corrective-action search space, the validation envelope that prevents the engine from exceeding the institution's risk appetite, and the regulatory pathway that documents the autonomous action with the same evidence chain that the current Regulatory Continuity Certificate provides for human-driven action (Wang et al., 2024; Gudipuri, 2026b).

Decentralised and atomic-settlement integration through modernized NonStop API layers would extend the framework's deployment to the post-trade-automation directions that the industry is pursuing under the T+1 and T+0 settlement horizons. The research agenda includes the multi-party clearing scenario in which the cryptographic anchoring of the Regulatory Continuity Certificate becomes part of the inter-counterparty trust chain (Byron, 2026).

Cross-jurisdictional regulatory harmonization of the Regulatory Continuity Certificate format would extend the supervisor-side acceptance of the Certificate from the three U.S. regimes currently addressed (SEC Rule 17a-4, FINRA Rule 4370, DTCC Member System

Change Notification) to a broader set of European, Asian, and other jurisdictions. The research agenda includes the formal mapping of the Certificate's internal structure to the jurisdiction-specific reporting templates and the supervisor-side cryptographic-verification primitive that the harmonisation requires.

The work invites continuation by practitioners in the institutions that operate HPE NonStop systems, by researchers whose interest lies in the fault-tolerant-systems literature, and by regulators whose remit includes the documentation regimes that the Regulatory Continuity Certificate is engineered to discharge. The patents that protect the framework's novel components are filed; the illustrative deployment model is documented; the implementation roadmap is documented; the limits, including the provisional-patent stage and the absence of an independent production deployment, are explicit; and the directions in which the work invites extension, foremost among them an independent production deployment with measured effects, are stated.

Further Reading

The Further Reading section organises the broader literature that informs the framework presented in this monograph. The entries below are arranged by thematic cluster; each entry appears in the Bibliography in alphabetical order. The clusters are intended to guide readers whose interest lies in a specific subdomain of the framework's intellectual context.

Large language models and code understanding

Foundational and recent surveys on large language models include Zhao et al. (2026) for a comprehensive review of the architecture and capabilities of modern models, Raiaan et al. (2024) for a structured taxonomy of model families and applications, and Wang et al. (2024) for an autonomous-agent perspective relevant to the AI Verification Engine's self-supervised behavior. Retrieval-augmented generation is the technique by which the Stage 1 Guardian Archaeology pass compensates for the absence of NonStop-specific material in the open training corpora; Gao et al. (2023) provide the canonical survey. Program understanding and code-generation literature relevant to the Shadow Code Generation stage includes Ahmad et al. (2021) on unified pre-training for program understanding, Park et al. (2023) on transformer-based code comment generation with aligned lexical attention, and Li et al. (2023, 2024) on AceCoder and effective

prompting techniques specialized for code generation. Static abstract-syntax-tree analysis, on which the Stage 1 TAL Analysis Pipeline relies, is treated by Antoniol et al. (2004) in their XOgastan XML-oriented AST analysis and transformation framework.

Microservice and service-oriented architectures

The framework's sidecar architecture (Stage 3, Patent #1) builds on the broader literature on microservice patterns, decomposition strategies, and service-mesh designs. Velepucha and Aguilar (2023) survey microservice principles, patterns, and migration challenges. Abgaz et al. (2023) review the systematic decomposition of monolith applications into microservices. Blinowski et al. (2022) compare monolithic and microservice architectures on performance and scalability dimensions. Hasan et al. (2023) measure architecture maintainability across the monolith-to-microservice transition. Ayas et al. (2023) report an empirical study of the systemic and technical migration toward microservices. Mehboob et al. (2025) discuss orchestration of distributed enterprises through microservices. Usman et al. (2022) survey the observability problem in distributed edge and container-based microservice systems. Reference architectures and tooling appear in Söylemez et al. (2023), Bogner et al. (2021) on industry practices for evolvability assurance, Alshuqayran et al. (2018) on architecture recovery, Soldani et al. (2021) on the μTOSCA toolchain, and Casale et al. (2019) on the RADON decomposition framework. NIST guidance on microservice security comes from Chandramouli (2019, 2020). Theodoropoulos et al. (2023) survey security in cloud-native services. Amiri et al. (2021) report on reliability-performance trade-offs in dynamic routing. Berardi et al. (2022) provide a systematic security review, and Casola et al. (2023) present a model-based methodology for secure software development and testing relevant to the sidecar attack surface. Monolith-decomposition strategies of the kind the API Gateway sidecar pattern presupposes are formalised by Mazlami et al. (2017) in their extraction of microservices from monolithic software architectures.

Fault-tolerant distributed systems and transaction processing

The framework's verification methodology (Stage 5, Patent #2) extends a deep literature on fault-tolerant distributed systems and transaction processing. Cristian (1991) establishes the foundational understanding of fault-tolerant distributed systems. Haerder and Reuter (1983)

define the principles of transaction-oriented database recovery on which the TMF audit trail rests. Mohan et al. (1986) report on transaction management in the R* distributed database system. Two-phase commit protocols are analysed in Stamos and Cristian (2002) on low-cost atomic commit, Boutros and Desai (2002) on protocol performance, and Lin et al. (2016) on non-2PC transaction management in distributed databases. Mahmoud et al. (2013) develop low-latency multi-datacenter databases using replicated commit. Mechanical verification of transaction processing systems and concurrency control is treated by Chkliaev et al. (2001, 2002). Modern consistency considerations appear in Dhanagari (2024) on MongoDB consistency under performance constraints, in Vogels (2008) on eventual consistency as a design point distinct from the strict consistency that TMF enforces, in Thomson and Abadi (2010) on the case for determinism in database systems, and in Manson et al. (2005) on the Java memory model, whose concurrency semantics the Stage 4 shadow code must respect when reproducing TMF-bracketed Enscribe access.

Legacy modernization and technical debt

The legacy-modernization literature that frames the NonStop problem includes Sneed (2020) on cost-driven software migration, Sneed (2019) on re-implementing a legacy system, Lucia et al. (2008) on legacy migration methods, and Brand and van der Brand (1997) for an annotated bibliography on reverse engineering and system renovation. Cipresso (2009) treats software reverse engineering education. Srinivas et al. (2016) provide a comparative survey of legacy systems. Mallidi et al. (2021) analyse TCO and ROI for legacy digital transformation. Kesharwani et al. (2025) develop an enterprise-scale cloud migration factory model. Melo et al. (2025) report on human-AI collaboration in COBOL modernization. Razavian and Lago (2015) survey service-oriented migration systematically. Lano et al. (2024) apply model-driven engineering to automated language translation. Kavianpour et al. (2007) cover SOA and large-scale enterprise transformation, and Papazoglou et al. (2007) provide the foundational service-oriented architectures survey. Technical-debt management literature includes Avgeriou et al. (2016) on the Dagstuhl seminar, Ciancarini et al. (2020) on the strategic technical debt management model, and Tamburri et al. (2015) on social debt in software engineering. Aalst (2013) surveys business process management, with relevance to the Stage 2 Intent Recognition methodology. The specific problem of automated language conversion, which the COBOL85-to-

JVM Shadow Code Generation stage confronts, is examined by Verhoef and Terekhov (2000) on the realities of language conversions, by Lämmel and Verhoef (2001) on the 500-language problem, by Lin (2008) on migrating to relational systems, by Mens and Tourwé (2004) in their survey of software refactoring, and by Ouni et al. (2017) on multi-objective refactoring recommendation.

Cloud-native banking, fintech, and financial-services modernization

The deployment context for NM4N intersects with the broader cloud-native banking and fintech modernization literature. Anusha (2025) presents a cloud-native modernization framework for core banking systems using AWS microservices. Challa (2021) develops cloud-native architecture for scalable fintech applications with real-time payments. Kruse et al. (2019) survey AI for the financial-services industry. Gozman et al. (2018) report on fintech innovation mechanisms through the SWIFT Innotribe competition. Cerny et al. (2020) discuss code-analysis opportunities and challenges for enterprise systems and microservices. Blanco et al. (2023) survey SaaS transformation in the automotive sector with structural lessons applicable to financial-services modernization.

Differential testing and equivalence checking

Mora et al. (2018) introduce client-specific equivalence checking, a methodology adjacent to the Stage 5 JVM Shadow Comparator's pairwise comparison logic. Noller et al. (2020) develop HyDiff for hybrid differential testing. The broader literature on regression testing, shadow execution, and equivalence checking provides the methodological context for the TMF-oracle approach, with the novel contribution of Patent #2 being the use of hardware-anchored audit records rather than synthetic test inputs as the ground-truth reference.

Glossary

A

- AL4 (Availability Level 4). The highest level of fault-tolerant operation in the HPE NonStop classification: no single processor, memory module, network adapter, or storage path constitutes a single point of failure.

C

- COBOL85 (NonStop variant). The HPE NonStop dialect of COBOL85, with platform-specific extensions for Guardian, TMF, and Enscribe interaction. Encodes business logic across most NonStop banking and telecom estates.

B

- Divergence Assessment Record. The structured output of the JVM Shadow Comparator for each correlated pair of legacy and shadow records, capturing the field-level differences and the LLM Fault Classifier's class assignment.

E

- Enscribe. The hierarchical record-oriented database of HPE NonStop. Supports key-sequenced, entry-sequenced, and relative file organizations; locking is implicit under TMF.

F

- Five-Class Fault Taxonomy. The classification of behavioral divergence between a legacy NonStop process and its JVM shadow into five classes: Class 0 (identical), Class 1 (acceptable formatting), Class 2 (business-logic divergence), Class 3 (TMF-committed, JVM no-acknowledgment, novel), Class 4 (JVM phantom commit, no TMF record, novel). Patent #2.

G

- Guardian. The operating system of HPE NonStop. Provides the message-based inter-process communication primitive (RECEIVE/REPLY) on which all higher-level NonStop services build.
- Guardian IPC Bridge. The Patent #1 interoperability layer that allows JVM services in the OSS environment to communicate with Guardian processes through native C/C++ Guardian system procedure calls and to participate in TMF distributed transactions.

J

- JVM Shadow Comparator. The Patent #2 component that correlates the TMF State Delta Record stream with the JVM Transaction Output Record stream by transaction identifier within a configurable time window and produces Divergence Assessment Records.

L

- LLM Fault Classifier. The fine-tuned large language model that assigns each Divergence Assessment Record to one of the five fault classes, with primary responsibility for the Class 1 versus Class 2 disambiguation.

M

- Modernization in place. The methodological principle of bringing modern capabilities to the legacy NonStop platform through additive sidecar layering rather than extracting workloads from the platform.

N

- NBPS (NonStop Business Process Specification). The structured output of Stage 2 Intent Recognition: a document that captures each business operation, the processes it traverses, the TMF transaction boundary, the Enscribe files it touches, the regulatory regime, and the human-facing description.
- NM4N (Neural Modernization for NonStop). The five-stage framework presented in this monograph for AI-driven modernization of HPE NonStop systems.

O

- OSS (Open System Services). The POSIX-compatible runtime environment that ships with HPE NonStop and shares its hardware. Hosts the sidecar services that attach to legacy NonStop processes through the Guardian IPC Bridge.

P

- Pathway. The NonStop transaction processing monitor. Coordinates Terminal Control Processes, server classes, and request distribution under fault-tolerant guarantees.
- Process pair. The HPE NonStop fault-tolerance primitive: a primary process and a backup process running in lockstep on separate processors, with state checkpointed from primary to backup at defined sync points; on primary failure, the backup transparently takes over.

R

- RCC (Regulatory Continuity Certificate). The Patent #2 cryptographically signed compliance artifact produced for each certification window: contains the divergence-class distribution, the TMF sequence-number range, and the attestation; archived to WORM storage suitable for SEC Rule 17a-4(f)(2)(ii)(A).

S

- SCREEN COBOL. The HPE NonStop dialect of COBOL for 6530-terminal screen handling under Pathway TCP.
- Sidecar. A modern service deployed in the OSS environment that attaches to legacy NonStop processes through Guardian IPC. Implements API gateway, stream processor, message adapter, or ML inference proxy patterns.
- SLO (Service Level Objective). The institutional commitment to a defined performance and availability level on the NonStop platform. The reference broker-dealer platform of Case Study B operates under a 99.97 percent performance SLO and a 100 percent availability SLO.
- SR 11-7. The U.S. Federal Reserve's Supervisory Letter 11-7 on Model Risk Management. Governs the validation and ongoing-monitoring requirements for models that bear on regulatory or capital decisions, including the LLM Fault Classifier.

T

- TACL (Tandem Advanced Command Language). The NonStop operator-interaction and batch-orchestration language. Provides macro-rich operational scripts.
- TAL (Tandem Application Language). The systems-implementation language of HPE NonStop. Provides direct calls to Guardian system procedures and the sublocal-variable construct for stack-persistent state across procedure calls. Approximately 0.001 percent of contemporary LLM training corpora.
- TMF (Transaction Management Facility). The HPE NonStop distributed-transaction coordinator. Writes hardware-anchored audit records to a durable, tamper-resistant audit trail BEFORE returning commit acknowledgment to the application process.

- TMF Harvest Engine. The Patent #2 Guardian-native daemon that reads the TMF audit trail in real time via the TMFCOM programmatic interface and emits State Delta Records keyed by TMF sequence number.
- TMF State Delta Record. The per-transaction record extracted by the TMF Harvest Engine: TMF sequence number, hardware-level commit timestamp, Enscribe file segments modified, record-level mutations, and process pair state at commit.

W

- WORM (Write Once Read Many). The archival storage pattern required by SEC Rule 17a-4(f)(2)(ii)(A) for non-alterable recordkeeping: data once written cannot be modified and remains readable indefinitely.

Bibliography

Aalst, W. M. P. v. d. (2013). Business Process Management: A Comprehensive Survey. ISRN Software Engineering, 2013, 1-37. https://doi.org/10.1155/2013/507984

Abgaz, Y., McCarren, A., Elger, P., Solan, D., Lapuz, N., Bivol, M., & ... (2023). Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review. IEEE Transactions on Software Engineering, 49(8), 4213-4242. https://doi.org/10.1109/tse.2023.3287297

Ahmad, W. U., Chakraborty, S., Ray, B., & Chang, K. (2021). Unified Pre-training for Program Understanding and Generation. https://doi.org/10.18653/v1/2021.naacl-main.211

Alshuqayran, N., Alí, N., & Evans, R. (2018). Towards Micro Service Architecture Recovery: An Empirical Study. https://doi.org/10.1109/icsa.2018.00014

Amiri, A., Zdun, U., & Hoorn, A. v. (2021). Modeling and Empirical Validation of Reliability and Performance Trade-Offs of Dynamic Routing in Service- and Cloud-Based Architectures. IEEE Transactions on Services Computing, 15(6), 3372-3386. https://doi.org/10.1109/tsc.2021.3098178

Antoniol, G., Penta, M. D., Masone, G., & Villano, U. (2004). XOgastan: XML-oriented gcc AST analysis and transformations. https://doi.org/10.1109/scam.2003.1238043

Anusha, A. (2025). Cloud-Native Modernization Framework for Core Banking Systems Using AWS Microservices. International Journal of Multidisciplinary Research and Growth Evaluation, 6(3), 1627-1634. https://doi.org/10.54660/.ijmrge.2025.6.3.1627-1634

Avgeriou, P., Kruchten, P., Özkaya, İ., & Seaman, C. (2016). Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162). DROPS (Schloss Dagstuhl – Leibniz Center for Informatics). https://doi.org/10.4230/dagrep.6.4.110

Ayas, H. M., Leitner, P., & Hebig, R. (2023). An empirical study of the systemic and technical migration towards microservices. Empirical Software Engineering, 28(4), 85. https://doi.org/10.1007/s10664-023-10308-9

Basel Committee on Banking Supervision. (2017, December 7). Basel III: Finalising post-crisis reforms (BCBS Publication d424). Bank for International Settlements. https://www.bis.org/bcbs/publ/d424.htm

Berardi, D., Giallorenzo, S., Mauro, J., Melis, A., Montesi, F., & Prandini, M. (2022). Microservice security: a systematic literature review. PeerJ Computer Science, 7, e779. https://doi.org/10.7717/peerj-cs.779

Blanco, D. F., Mouël, F. L., Lin, T., & Escudié, M. (2023). A Comprehensive Survey on Software as a Service (SaaS) Transformation for the Automotive Systems. IEEE Access, 11, 73688-73753. https://doi.org/10.1109/access.2023.3294256

Blinowski, G., Ojdowska, A., & Przybyłek, A. (2022). Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. IEEE Access, 10, 20357-20374. https://doi.org/10.1109/access.2022.3152803

Board of Governors of the Federal Reserve System & Office of the Comptroller of the Currency. (2011, April 4). Supervisory Guidance on Model Risk Management (SR Letter 11-7; OCC Bulletin 2011-12). Federal Reserve System & U.S. Department of the Treasury.

Bogner, J., Fritzsch, J., Wagner, S., & Zimmermann, A. (2021). Industry practices and challenges for the evolvability assurance of microservices. Empirical Software Engineering, 26(5). https://doi.org/10.1007/s10664-021-09999-9

Boutros, B., & Desai, B. C. (2002). A two-phase commit protocol and its performance. https://doi.org/10.1109/dexa.1996.558282

Brand, M. v. d., Klint, P., & Verhoef, C. (1997). Reverse engineering and system renovation-an annotated bibliography. ACM SIGSOFT Software Engineering Notes, 22(1), 57-68. https://doi.org/10.1145/251759.251849

Byron, S. (2026, April 7). AI + digital assets: The next operational frontier for financial markets [Blog post]. Securities Industry and Financial Markets Association. https://www.sifma.org/news/blog/ai-digital-assets-the-next-operational-frontier-for-financial-markets

Casale, G., Artač, M., Heuvel, W. v. d., Hoorn, A. v., Jakovits, P., Leymann, F., & ... (2019). RADON: rational decomposition and orchestration for serverless computing. SICS Software-Intensive Cyber-Physical Systems, 35(1-2), 77-87. https://doi.org/10.1007/s00450-019-00413-w

Casola, V., De Benedictis, A., Rak, M., & Villano, U. (2023). Secure software development and testing: A model-based methodology. Computers & Security, 132, Article 103639. https://doi.org/10.1016/j.cose.2023.103639

Cerny, T., Svacina, J., Das, D., Bushong, V., Bureš, M., Tisnovsky, P., & ... (2020). On Code Analysis Opportunities and Challenges for Enterprise Systems and Microservices. IEEE Access, 8, 159449-159470. https://doi.org/10.1109/access.2020.3019985

Challa, K. (2021). Cloud Native Architecture for Scalable Fintech Applications with Real Time Payments. International Journal Of Engineering And Computer Science, 10(12), 25501-25515. https://doi.org/10.18535/ijecs.v10i12.4654

Chandramouli, R. (2019). Security strategies for microservices-based application systems. https://doi.org/10.6028/nist.sp.800-204

Chandramouli, R., & Butcher, Z. (2020). Building secure microservices-based applications using service-mesh architecture. https://doi.org/10.6028/nist.sp.800-204a

Chkliaev, D. D. (2001). Mechanical verification of concurrency control and recovery protocols. Data Archiving and Networked Services (DANS). https://doi.org/10.6100/ir547956

Chkliaev, D., Hooman, J., & Stok, P. v. d. (2002). Mechanical verification of transaction processing systems. https://doi.org/10.1109/icfem.2000.873809

Ciancarini, P., Farina, M., Masyagin, S., Succi, G., Yermolaieva, S., & Zagvozkina, N. (2020). The strategic technical debt management model: An empirical proposal. In Open Source Systems (Lecture Notes in Computer Science, Vol. 556, pp. 131-140). Springer. https://doi.org/10.1007/978-3-030-47240-5_13

Cipresso, T. (2009). Software reverse engineering education. https://doi.org/10.31979/etd.4ppy-2cjg

Cristian, F. (1991). Understanding fault-tolerant distributed systems. Communications of the ACM, 34(2), 56-78. https://doi.org/10.1145/102792.102801

Depository Trust and Clearing Corporation. (2024). Member System Change Notification Requirements [Member services documentation]. DTCC.

Dhanagari, M. R. (2024). MongoDB and Data Consistency: Bridging the Gap between Performance and Reliability. Journal of Computer Science and Technology Studies, 6(2), 183-198. https://doi.org/10.32996/jcsts.2024.6.2.21

Euroclear. (2024). Annual report 2024: Settlement, custody, and collateral services. Euroclear Group.

European Parliament and Council. (2016, April 27). Regulation (EU) 2016/679 of the European Parliament and of the Council on the protection of natural persons with regard to the processing of personal data (General Data Protection Regulation). Official Journal of the European Union, L 119, 1-88. https://eur-lex.europa.eu/eli/reg/2016/679/oj

European Parliament and Council. (2022, December 14). Regulation (EU) 2022/2554 on digital operational resilience for the financial sector (Digital Operational Resilience Act). Official Journal of the European Union, L 333, 1-79. https://eur-lex.europa.eu/eli/reg/2022/2554/oj

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., & ... (2020). CodeBERT: A Pre-Trained Model for Programming and Natural Languages. https://doi.org/10.18653/v1/2020.findings-emnlp.139

Financial Industry Regulatory Authority. (2004, as amended 2015). FINRA Rule 4370: Business Continuity Plans and Emergency Contact Information. FINRA Rulebook. https://www.finra.org/rules-guidance/rulebooks/finra-rules/4370

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., & ... (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2312.10997

Gozman, D., Liebenau, J., & Mangan, J. (2018). The Innovation Mechanisms of Fintech Start-Ups: Insights from SWIFT's Innotribe Competition. Journal of Management Information Systems, 35(1), 145-179. https://doi.org/10.1080/07421222.2018.1440768

Gudipuri, N. (2026a). System and method for bridging Java Virtual Machine services with HPE NonStop Guardian operating system processes while preserving fault-tolerant transaction semantics [U.S. Provisional Patent Application No. 63/990,182]. United States Patent and Trademark Office.

Gudipuri, N. (2026b). TMF-anchored transactional divergence detection and regulatory continuity certification during incremental JVM migration on HPE NonStop Guardian architectures [U.S. Provisional Patent Application No. 64/044,670; USPTO Confirmation No. 6319; filed 2026-04-20]. United States Patent and Trademark Office.

Haerder, T., & Reuter, A. (1983). Principles of transaction-oriented database recovery. ACM Computing Surveys, 15(4), 287-317. https://doi.org/10.1145/289.291

Hasan, M. H., Osman, M. H., Admodisastro, N., & Muhammad, S. (2023). From Monolith to Microservice: Measuring Architecture Maintainability. International Journal of Advanced Computer Science and Applications, 14(5). https://doi.org/10.14569/ijacsa.2023.0140591

Hewlett Packard Enterprise. (2024a). HPE NonStop Operating System Guardian Programmer's Guide [Technical documentation]. HPE Support Center.

Hewlett Packard Enterprise. (2024b). HPE NonStop Transaction Management Facility (TMF) Programmer's Guide [Technical documentation]. HPE Support Center.

Hewlett Packard Enterprise. (2024c). HPE NonStop Pathway/iTS System Management Manual [Technical documentation]. HPE Support Center.

Hewlett Packard Enterprise. (2024d). Enscribe Programmer's Guide [Technical documentation]. HPE Support Center.

Kavianpour, M., Williams, J., Strang, K., & Whitesides, J. (2007). SOA and large-scale and complex enterprise transformation. In Service-Oriented Computing (ICSOC 2007) (Lecture Notes in Computer Science, Vol. 4749, pp. 757-768). Springer. https://doi.org/10.1007/978-3-540-74974-5_50

Kesharwani, A., Manker, L., & Rajput, A. (2025). Enterprise-Scale Cloud Migration Factory Model: A Standardized and Value-Driven Approach for Large-Scale Applications. Computer Science and Engineering, 15(1), 22-35. https://doi.org/10.5923/j.computer.20251501.03

Kruse, L., Wunderlich, N., & Beck, R. (2019). Artificial Intelligence for the Financial Services Industry: What Challenges Organizations to Succeed. Proceedings of the ... Annual Hawaii International Conference on System Sciences/Proceedings of the Annual Hawaii International Conference on System Sciences. https://doi.org/10.24251/hicss.2019.770

Lano, K., & Siala, H. A. (2024). Using model-driven engineering to automate software language translation. Automated Software Engineering, 31(1). https://doi.org/10.1007/s10515-024-00419-y

Li, J., Zhao, Y., Li, Y., Li, G., & Jin, Z. (2023). AceCoder: Utilizing Existing Code to Enhance Code Generation. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2303.17780

Lin, C. (2008). Migrating to Relational Systems: Problems, Methods, and Strategies. Contemporary Management Research, 4(4). https://doi.org/10.7903/cmr.1127

Lin, Q., Chang, P., Chen, G., Ooi, B. C., Tan, K., & Wang, Z. (2016). Towards a Non-2PC Transaction Management in Distributed Database Systems. https://doi.org/10.1145/2882903.2882923

Lucia, A. D., Francese, R., Scanniello, G., & Tortora, G. (2008). Developing legacy system migration methods and tools for technology transfer. Software Practice and Experience, 38(13), 1333-1364. https://doi.org/10.1002/spe.870

Lämmel, R., & Verhoef, C. (2001). Cracking the 500-language problem. IEEE Software, 18(6), 78-88. https://doi.org/10.1109/52.965809

Mahmoud, H. A., Nawab, F., Pucher, A., Agrawal, D., & Abbadi, A. E. (2013). Low-latency multi-datacenter databases using replicated commit. Proceedings of the VLDB Endowment, 6(9), 661-672. https://doi.org/10.14778/2536360.2536366

Mallidi, R. K., Sharma, M., & Singh, J. (2021). Legacy Digital Transformation: TCO and ROI Analysis. International journal of electrical and computer engineering systems, 12(3), 163-170. https://doi.org/10.32985/ijeces.12.3.5

Manson, J., Pugh, W., & Adve, S. V. (2005). The Java memory model. https://doi.org/10.1145/1040305.1040336

Mazlami, G., Cito, J., & Leitner, P. (2017). Extraction of Microservices from Monolithic Software Architectures. https://doi.org/10.1109/icws.2017.61

Mehboob, S. A. (2025). Orchestrating the Distributed Enterprise: Microservices as Catalysts for Systems Integration Evolution. International Journal of Small Business and Entrepreneurship Research, 13(1), 73-88. https://doi.org/10.37745/ijsber.2013/vol13n17388

Melo, I. E. S. d., Polónia, D. F., & Teixeira, L. (2025). Human–AI Collaboration in the Modernization of COBOL-Based Legacy Systems: The Case of the Department of Government Efficiency (DOGE). Computers, 14(7), 244. https://doi.org/10.3390/computers14070244

Mens, T., & Tourwé, T. (2004). A survey of software refactoring. IEEE Transactions on Software Engineering, 30(2), 126-139. https://doi.org/10.1109/tse.2004.1265817

Mohan, C., Lindsay, B. G., & Obermarck, R. (1986). Transaction management in the R* distributed database management system. ACM Transactions on Database Systems, 11(4), 378-396. https://doi.org/10.1145/7239.7266

Mora, F., Li, Y., Rubin, J., & Chećhik, M. (2018). Client-specific equivalence checking. https://doi.org/10.1145/3238147.3238178

Noller, Y., Păsăreanu, C. S., Böhme, M., Sun, Y., Nguyen, H. L., & Grunske, L. (2020). HyDiff: Hybrid differential software analysis. Proceedings of the 42nd International Conference on Software Engineering (ICSE 2020), 1273-1285. https://doi.org/10.1145/3377811.3380363

Ouni, A., Kessentini, M., Cinnéide, M. Ó., Sahraoui, H., Deb, K., & Inoue, K. (2017). MORE: A multi-objective refactoring recommendation approach to introducing design patterns and fixing code smells. Journal of Software Evolution and Process, 29(5). https://doi.org/10.1002/smr.1843

Papazoglou, M. P., & Heuvel, W. v. d. (2007). Service oriented architectures: approaches, technologies and research issues. The VLDB Journal, 16(3), 389-415. https://doi.org/10.1007/s00778-007-0044-3

Park, Y., Park, A., & Kim, C. (2023). ALSI-Transformer: Transformer-Based Code Comment Generation With Aligned Lexical and Syntactic Information. IEEE Access, 11, 39037-39047. https://doi.org/10.1109/access.2023.3268638

Raiaan, M. A. K., Mukta, M. S. H., Fatema, K., Fahad, N. M., Sakib, S., Mim, M. M. J., & ... (2024). A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges. IEEE Access, 12, 26839-26874. https://doi.org/10.1109/access.2024.3365742

Razavian, M., & Lago, P. (2015). A systematic literature review on SOA migration. Journal of Software Evolution and Process, 27(5), 337-372. https://doi.org/10.1002/smr.1712

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. https://doi.org/10.18653/v1/d19-1410

Sneed, H. M. (2019). Re-implementing a legacy system. Journal of Systems and Software, 155, 243-262. https://doi.org/10.1016/j.jss.2019.05.012

Sneed, H. M., & Verhoef, C. (2020). Cost-driven software migration: An experience report. Journal of Software Evolution and Process, 32(7). https://doi.org/10.1002/smr.2236

Soldani, J., Muntoni, G., Neri, D., & Brogi, A. (2021). The μTOSCA toolchain: Mining, analyzing, and refactoring microservice-based architectures. Software Practice and Experience, 51(7), 1591-1621. https://doi.org/10.1002/spe.2974

Srinivas, M., Ramakrishna, G., Rao, K. R., & Babu, E. S. (2016). Analysis of Legacy System in Software Application Development: A Comparative Survey. International Journal of Electrical and Computer Engineering (IJECE), 6(1), 292. https://doi.org/10.11591/ijece.v6i1.pp292-297

Stamos, J. W., & Cristian, F. (2002). A low-cost atomic commit protocol. https://doi.org/10.1109/reldis.1990.93952

Söylemez, M., Tekinerdoğan, B., & Tarhan, A. (2023). Microservice reference architecture design: A multi-case study. Software Practice and Experience, 54(1), 58-84. https://doi.org/10.1002/spe.3241

Tamburri, D. A., Kruchten, P., Lago, P., & Vliet, H. v. (2015). Social debt in software engineering: insights from industry. Journal of Internet Services and Applications, 6(1). https://doi.org/10.1186/s13174-015-0024-6

Theodoropoulos, T., Rosa, L., Benzaïd, C., Gray, P., Marin, E., Makris, A., Cordeiro, L., Diego, F., Sotiriou, M., Tabatabaee Malazi, P., Kor-Hadji-Vasilev, V., & Tserpetsas, A. (2023). Security in cloud-native services: A survey. Journal of Cybersecurity and Privacy, 3(4), 758-793. https://doi.org/10.3390/jcp3040034

Thomson, A., & Abadi, D. J. (2010). The case for determinism in database systems. Proceedings of the VLDB Endowment, 3(1-2), 70-80. https://doi.org/10.14778/1920841.1920855

U.S. Securities and Exchange Commission. (1997, as amended). Rule 17a-4, Records to be preserved by certain exchange members, brokers, and dealers (17 C.F.R. § 240.17a-4). Electronic Code of Federal Regulations.

Usman, M., Ferlin, S., Brunström, A., & Taheri, J. (2022). A Survey on Observability of Distributed Edge & Container-Based Microservices. IEEE Access, 10, 86904-86919. https://doi.org/10.1109/access.2022.3193102

Velepucha, V., & Flores, P. (2023). A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges. IEEE Access, 11, 88339-88358. https://doi.org/10.1109/access.2023.3305687

Verhoef, C., & Terekhov, A. (2000). The realities of language conversions. IEEE Software, 17(6), 111-124. https://doi.org/10.1109/52.895180

Vogels, W. (2008). Eventually consistent. Communications of the ACM, 52(1), 40-44. https://doi.org/10.1145/1435417.1435432

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., & ... (2024). A survey on large language model based autonomous agents. Frontiers of Computer Science, 18(6). https://doi.org/10.1007/s11704-024-40231-1

Zhao, P., Wu, X., Li, Z., & Zhao, J. (2023). QChecker: Detecting Bugs in Quantum Programs via Static Analysis. https://doi.org/10.1109/q-se59154.2023.00014

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., & ... (2026). A Survey of Large Language Models. Frontiers of Computer Science, 20(12). https://doi.org/10.1007/s11704-026-60308-3