publicis
sapient

How Regulated Enterprises Modernize Legacy Systems Safely

Real case studies showing how AI reduced risk and increased speed for financial services, healthcare, energy and commodities industries

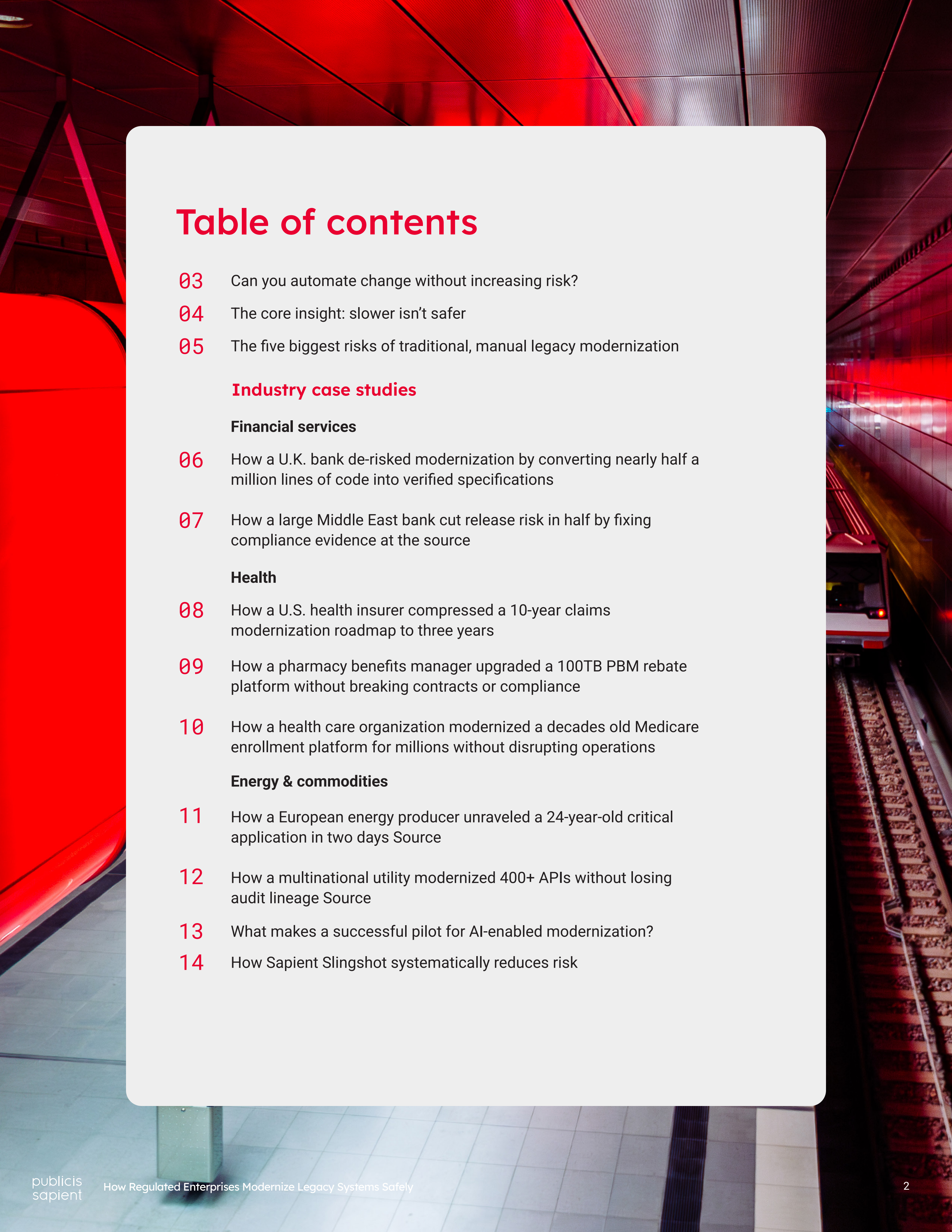


Table of contents

Can you automate change without increasing risk?

Regulated organizations do not delay modernization because they lack ambition. They delay because failure carries severe consequences: regulatory findings, customer or member harm, security exposure and operational disruption.

In financial services, health and energy, legacy modernization is not just a technical exercise. Teams must also be able to prove—to auditors, regulators and internal risk committees—that system behavior, data handling and controls remain intact after change. Speed without proof increases risk. And historically, proof has been slow, manual and fragile.

This playbook documents how large, regulated enterprises can modernize mainframe and legacy application platforms with less risk for the first time with the help of AI.

The cases below are based on real-world, AI-enabled modernization programs where regulatory scrutiny was constant, and tolerance for error was low.

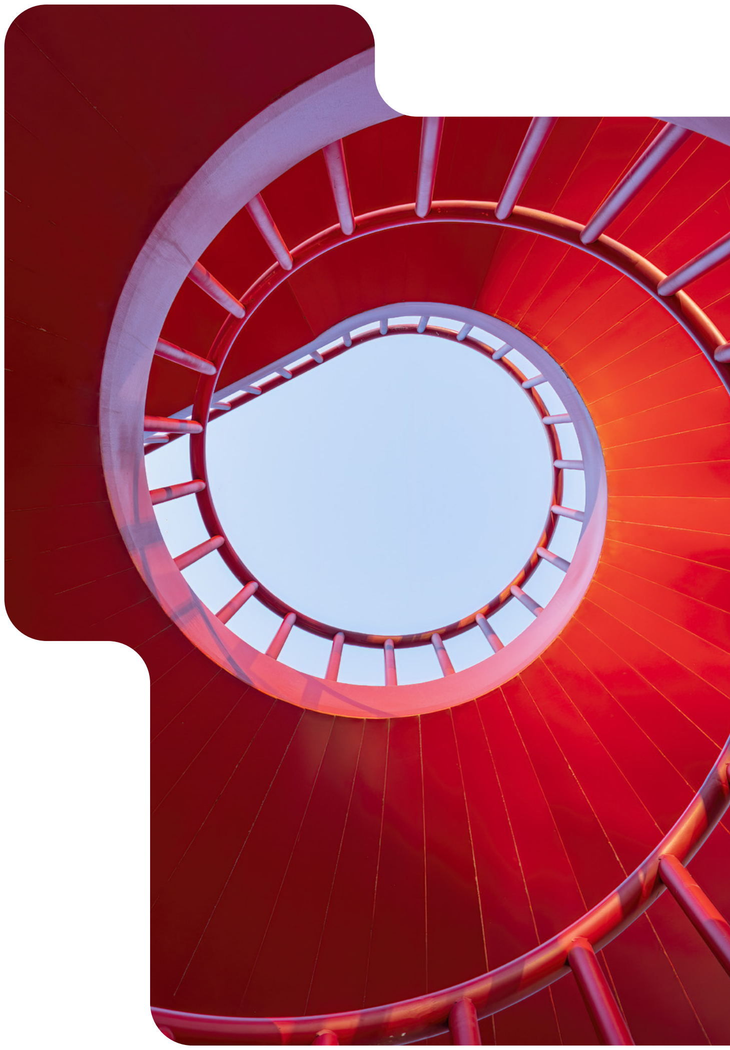
The core insight: slower isn’t safer

Across every story, risk went down when large technology systems became more observable, more testable and more governable before change.

AI’s purpose in modernization was not just “coding faster.”

Its true value was:

- Making hidden behavior explicit
- Proving equivalence continuously
- Generating audit-ready evidence as part of delivery



The five biggest risks of traditional, manual legacy modernization

In each case, you'll see that AI was applied specifically to reduce one or more of the five biggest real-world factors threatening large enterprises during a "status quo" legacy modernization:

01 Extended timelines increased exposure

Partially modernized systems remain in production too long, prolonging regulatory exposure and delaying return on investment

02 Undocumented dependencies caused downstream failures

Hidden system and data dependencies trigger outages, rollbacks and loss of confidence in future change

03 Security and data-handling exposure

Refactoring code introduces new vulnerabilities, expanding audit scope and regulatory exposure

04 Lack of audit-grade traceability

Teams cannot demonstrate how requirements, code and tests align, forcing manual reconstruction of compliance evidence

05 Unintended rule changes

Regulated business rules are reimplemented without shared understanding of the real-world system logic, leading to unintended changes in claims, payments, coverage or eligibility

For each industry, reducing these risks through governed automation delivered measurable business outcomes: faster returns on new technology, reduced subject-matter expert (SME) dependency and millions in avoided technical debt. This playbook highlights how **Sapient Slingshot,** Publicis Sapient's enterprise AI platform for software development, made these outcomes possible and how executives should scope pilot projects in their organization to do the same.

Industry case study 1: financial services

U.K. retail and commercial bank

Converted half a million lines of code to verified specifications in eight weeks

Industry context

The company: Large U.K. retail and commercial bank modernizing legacy banking services under regulatory oversight

What was at stake

Core services are classified as "important business services" under U.K. operational resilience requirements. A small defect introduced during modernization that affects reporting, payments or customer processing could trigger:

- Supervisory intervention
- Formal remediation programs
- Board-level escalation
- Capital impact assessments
- Reputational damage

Status quo approach

Traditional code-to-spec efforts rely heavily on manual, human analysis and SMEs. This approach is slow, inconsistent and very difficult to scale. Specifications (i.e., descriptions of what a system is supposed to do) in large enterprises are often incomplete, out-of-date or disconnected from the underlying code, increasing the likelihood of missed logic and rework during later phases of the program, as well as the risks stated above from slowing down the initiative.

Why AI was necessary

The bank could not safely modernize ~500,000 lines of code tightly coupled with mainframe logic using manual interpretation without risking unintended rule changes or increased compliance exposure.

AI-enabled modernization approach

In this case, Sapient Slingshot, Publicis Sapient's enterprise AI software development platform, was applied specifically to extract the rules and behavior buried in code, and document them clearly. Because of this, the bank was able to:

1. Restore visibility before change

The platform rapidly analyzed a large volume of existing production code to extract hidden business logic and automatically generate traceable, reviewable specifications.

2. Establish understanding of system behavior

In just eight weeks, Slingshot drove code-to-spec efforts for 237 programs and 327 feeds, amounting to nearly half a million lines of code across mainframe batch feeds for financial and data products, along with the bank's payments module.

3. Design from validated specifications

Slingshot produced business specifications, designed a modern target-state architecture and revamped the underlying data model.

Outcomes

- **50%** faster verified specification creation
- **70–85%** reduction in manual code-to-spec effort (35 days → 5 days)
- **95%** specification accuracy
- Explicit traceability between legacy code and generated specifications
- Reduced SME dependency

Industry case study 2: financial services

Large Middle East bank

Stabilized 30+ systems and cut release risk under regulatory scrutiny

Industry context

The company: Large Gulf-region retail and commercial bank operating under regional banking oversight

What was at stake

A vendor's exit left the bank with 30+ fragmented services, a weak security posture and less than five percent of planned scope delivered—all under the scrutiny of regional banking regulators. Long release cycles risked turning fixable defects into regulatory events, and a months-long remediation gap is exactly what regulators would flag as "failure to remediate a control weakness in a timely manner," triggering:

- Regulatory capital impact
- Remediation program costs
- Restriction on new digital releases
- Required board-level reporting

Status quo approach

Without automation, modernizing 30+ interconnected banking services is linear, manual and prone to human error.

1. It starts with manual reverse engineering of code, architecture discovery done by human experts, manual backlog rebuilds and tech debt compounding.
2. The team, scope and costs expand, increasing integration defects.
3. At the end of project, teams can't guarantee "doneness," and risk slightly out-of-sync, crucial systems that are still reliant on the legacy core.

Why AI was necessary

Manual modernization of 30+ interdependent services made it impossible to move faster and still maintain strong oversight.

AI-enabled modernization approach

Implementing Sapient Slingshot, Publicis Sapient's enterprise AI software development platform, the bank was able to:

1. Stabilize all legacy services

Identify design flaws, improve architectural visibility, map and clarify service dependencies and reduce defect carryover

2. Increase test coverage and control maturity

Achieve 80 percent+ unit test coverage, automate regression generation, reduce defect rate by 30 percent and enforce compliance and security rules.

3. Compress review and release time

Shorten review cycles, streamline approval cycles, reduce testing bottlenecks and improve traceability across the whole lifecycle.

4. Improve delivery efficiency

Streamline the backlog-to-deployment cycle, meaning less rework and fewer manual bottlenecks.

Outcomes

- **30+** systems stabilized
- Unit test coverage increased to **80%+**
- **50%** faster review and release cycles
- **30%** defect reduction

Industry case study 3: health

U.S. health insurance company

Compressed 10-year claims modernization to ~three years without compliance drift

Industry context

The company: Leading U.S. health insurer processing billions of claims annually under CMS and HIPAA oversight

What was at stake

Claims adjudication logic was embedded across 10,000+ pages of COBOL on a legacy mainframe. At this scale, minor defects can trigger major compliance events—improper denials, provider underpayments or PHI exposure.

Potential consequences included:

- CMS audit findings
- Mandatory reprocessing
- Public reporting
- Class action lawsuits
- Congressional inquiry

Why AI was necessary

Proving that the new system behaves the same way as the old system (behavioral equivalence) across 10,000+ pages was necessary, yet infeasible through manual review alone.

Status quo approach

Traditional modernization focuses on rewriting code—not proving equivalence. After three years, only ~10 percent of features had been converted manually.

Risks included:

- Screen-by-screen rewrites prone to rule drift
- Business logic trapped in code and tribal knowledge
- SME dependency creating operational fragility
- Point-in-time testing that risked PHI exposure and HIPAA breach

This forced a defensive delivery posture: Teams slowed progress to reduce error risk, extending the modernization timeline and prolonging exposure.

AI-enabled modernization approach

By implementing [Sapient Slingshot, Publicis Sapient's enterprise AI software development platform, the organization was able to:]()

1. Complete discovery and rule extraction before change

- Automatically deconstruct legacy COBOL programs
- Extract embedded business logic into structured specifications
- Derive functional requirements directly from production behavior
- Reduce reliance on tribal knowledge

2. Phase modernization with continuous validation

Modern Java and React microservices were generated from validated specifications.

For every migrated feature, the team:

- Generated manual and automated test cases
- Ran regression comparisons between legacy and modern outputs
- Validated behavior against production data

No feature was complete without proven behavioral parity.

3. Embed security and control into delivery

- Replaced insecure legacy patterns during extraction
- Implemented cloud-native security frameworks
- Delivered full business logic documentation and traceability

Outcomes

- Modernization reduced from seven to 10 years to ~three years
- **$90M** budget reduction
- Significant reduction in inherited legacy vulnerabilities
- Full system-to-business logic traceability
- Reduced SME dependency

Industry case study 4: health

U.S. pharmacy benefits manager

Modernized 100TB financial system in under three years without breaking contracts or compliance

Industry context

The company: Large U.S. PBM managing billions in rebates and federal/state reporting requirements

What was at stake

The risks were technical and financial:

- A legacy rebate engine encoding decades of pricing and contract logic
- 100TB of rebate, pricing and market-share rule data
- Contractual exposure: Hundreds of manufacturer agreements with quarterly term changes
- CMS reporting dependent on exact preservation of financial calculations

A misapplied contract term or subtle shift in eligibility logic could distort an entire quarter's invoicing. A downstream allocation error could affect thousands of employer groups. The organization could not afford unintended rule changes and a five-to seven-year, dual-platform transition.

Why AI was necessary

Manual lifecycle reconstruction of rebate logic across 100TB of data was operationally impractical.

Status quo approach

- Contract clauses were embedded across services, batch jobs, and stored procedures
- More than 50 financial reports depended on precise upstream rebate calculations
- SMEs were required to manually validate accrual logic each quarter
- Regression testing required full financial output reconciliation

AI-enabled modernization approach

Sapient Slingshot, Publicis Sapient's enterprise AI platform for software development, did the opposite:

1. Discovery and rebate rule extraction complete up-front

- AI analyzed the full rebate calculation flow from pricing reference data through accrual and invoice generation
- Embedded manufacturer contract clauses extracted into structured, reviewable financial specifications
- Cross-program differences (Commercial vs. Medicaid) explicitly mapped

2. Financial lifecycle sequencing

- Modernization phases aligned directly to financial calculation dependencies
- Migration sequenced by lifecycle domain: pricing → contracts → claims → accrual → invoicing → reporting
- Each domain validated against legacy invoice and accrual outputs before advancing

3. Financial regression and governance

- Automated regression suites generated to cover tier thresholds, pricing overrides and contractual edge cases
- Every extracted financial rule traced from legacy source to modern implementation
- Validation artifacts produced continuously, not reconstructed post-audit
- No calculation domain migrated without proven invoice and accrual equivalence across representative production datasets

Outcomes

- Timeline reduced from five to seven years to ~two and a half years
- **50%** reduction in SME validation effort
- Full audit-ready rule documentation
- System behavior stayed consistent during parallel operation
- Eliminated dual-platform financial reconciliation risk

Industry case study 5: health

Medicare enrollment platform

Modernized a critical Medicare enrollment platform without risking coverage loss

Industry context

The company: U.S. health care organization managing Medicare enrollment and billing

What was at stake

The risks were technical, financial and regulatory:

- Technology lifecycle risk: A legacy eligibility platform encoding complex CMS enrollment rules.
- Data-scale risk: Decades of beneficiary enrollment and billing data.
- Regulatory risk: Coverage errors could trigger CMS findings and member disruption.
- Regulatory risk: CMS and state reporting dependent on precise upstream calculations

A subtle eligibility logic shift could result in wrongful coverage termination.

The organization could not risk unintended rule changes — or a five- to seven-year dual-platform transition.

Why AI was necessary

Manual reconstruction of enrollment and eligibility dependencies risked coverage gaps at scale.

Status quo approach

- Eligibility and billing rules were embedded across interdependent systems
- Coverage logic relied on undocumented rule interactions
- SMEs were required to validate enrollment outcomes manually
- Full rebate lifecycle regression testing was highly complex
- Regression testing required member-level coverage reconciliation

AI-enabled modernization approach

Sapient Slingshot was deployed inside the client's regulated environment, integrated with their AI Studio and certified LLM models, enabling:

1. Eligibility and workflow discovery

- Discovery and enrollment rule extraction complete up-front
- AI analyzed the full eligibility workflow — intake through coverage determination and premium billing
- Embedded CMS eligibility logic extracted into structured, reviewable specifications

Plan-level and regulatory rule differences explicitly mapped

2. Enrollment workflow sequencing

- Modernization phases aligned directly to operational workflow dependencies
- Migration sequenced by lifecycle domain: intake → eligibility → billing → reporting
- Each domain validated against legacy member-level coverage and billing outcomes before advancing

3. Behavioral regression and governance

- Automated regression suites generated to detect eligibility drift and billing inconsistencies
- Every rule traced from legacy source to modern implementation
- Validation artifacts generated continuously, not reconstructed post-audit
- No workflow migrated without proven behavioral equivalence across representative production datasets

Outcomes

- **30–40%** automation in rebuild
- Coverage integrity preserved for millions
- CMS reporting continuity maintained
- Predictable phased roadmap created
- Reduced risk of eligibility and billing defects

Industry case study 6: energy and commodities

Energy infrastructure

Revived a 25-year-old black-box application without rebuilding it from scratch

Industry context

The company: European energy producer operating gas and generation assets where system reliability directly impacts operational continuity and financial performance

What was at stake

For a European energy producer, modernization missteps in core operational systems didn't just risk project failure—they risked grid stability, regulatory compliance and continuity of service. If an application fails or remains out-of-date, an energy organization might face:

- Potential financial loss from slowed generation ramp-up
- Escalating security and compliance exposure from running unpatchable legacy code

However, rebuilding an application from scratch without fully understanding its original business logic risks recreating the same errors.

Why AI was necessary

Traditional modernization would require weeks of manual reverse engineering by senior engineers with no ability for leaders to measure success or completeness.

Status quo approach

This application exists only as compiled binaries, basically machine code (binary instructions made of 0s and 1s) that a computer's CPU can execute directly. Therefore:

- Any logic behind the code can't be read or validated
- The code can't run reliably on modern machines
- Code testing and validation can't be improved, because they happen at the source level
- Binary code does not preserve enough structure to make security updates

AI-enabled modernization approach

The team applied a structured five-step process accelerated by Sapient Slingshot, Publicis Sapient's enterprise AI platform for software development.

1. Decompilation and recovery

Using open-source AI tools, binary files were converted back into readable Java source code to restore the foundation for modernization.

2. Environment rebuild

A modern runtime environment (Java 17 and PostgreSQL 16) was created so the application could operate on current systems.

3. Refactoring and cleanup

Sapient Slingshot was used to restructure and modernize the recovered codebase, reducing over 9,000 lines to approximately 5,000 clean, readable lines with updated syntax and standards.

4. Business logic extraction

AI automatically generated entity-relationship diagrams and data-flow mappings, revealing the application's functional logic, something no one in the entire company could previously access.

5. Documentation and testability

Inline documentation and standalone artifacts were automatically generated, transforming the app from opaque code into a maintainable system.

Outcomes

- Modernization completed in two days instead of weeks
- 30–40% efficiency gains in test creation and automated code restructuring
- Legacy black-box system converted into readable, maintainable source code
- Operational continuity risk materially reduced
- Security and upgradeability restored
- Application now deployable across additional generation sites

Industry case study 7: energy and commodities

Energy & utilities

Migrated more than 400 APIs without losing regulatory lineage

Industry context

The company: A multinational energy and utilities company operating regulated electricity and gas infrastructure in the United States is subject to strict federal and state oversight, including:

- NERC CIP cybersecurity standards
- FERC reliability requirements
- Public utility commission reporting mandates

In this environment, system traceability, data integrity and operational continuity are regulatory obligations — not business preferences.

What was at stake

The company relied on a large, aging API estate to move operational and renewable-generation data across IT and OT systems supporting grid operations, renewable asset ingestion, regulatory reporting and security workflows.

Modernization carried significant risk: breaking generation and reporting integrations, losing required audit lineage and weakening visibility into how regulated data was transformed.

The organization needed to scale renewable data ingestion without increasing compliance exposure.

Why AI was necessary

The volume of APIs and interconnections made it impractical to manually trace data origins, transformations and dependencies across regulated systems.

Status quo approach

In large enterprises, APIs are deeply embedded across legacy systems, reports and partner integrations—often with incomplete documentation. Changes require manual coordination and heavy oversight to avoid disruption. However, a manual approach often means:

- Limited visibility into upstream and downstream dependencies
- Unpredictable cross-system impacts
- Reconstruction of compliance evidence after changes
- Delayed modernization due to regulatory risk concerns

AI-enabled modernization approach

The organization applied a modernization model supported by Sapient Slingshot, Publicis Sapient's enterprise AI platform for software development, that ensured:

1. A defined scope for the project

A bounded API domain supporting regulated operational data flows was selected.

2. Controls before change

- Inventory of system interdependencies established
- Data origin and transformation paths formally documented
- Scalable system modernization workflow established

3. Governed migration

- More than 400 APIs migrated from MuleSoft to Azure API Management
- Legacy code automatically converted into optimized Java services
- Built-in "paper trail" generated as part of delivery
- New integrations include a validated dependency impact assessment

Outcomes

- More than 400 APIs migrated
- ROI achieved within one year
- Eight figure projected ROI increase within three years
- None of the system connections that are subject to regulatory oversight were disrupted or broken
- Audit evidence generated continuously

What makes a successful pilot for AI-enabled modernization?

In every case documented in this playbook—whether it was a bank, health insurance company or energy producer—the organization began with a deliberately constrained pilot designed to reduce risk before increasing speed:

01 Pilot scope is intentionally narrow

A pilot for AI-enabled legacy modernization should focus on:

- A single regulated journey, domain or system slice (for example: a claims flow, billing module, API domain or mainframe program cluster)
- A bounded time frame, often two to four weeks or less
- No requirement to change production behavior to begin
- Limiting blast radius and making outcomes easy to evaluate

02 Controls are established before code changes

Before any code refactoring or platform migration:

- Existing business logic is extracted into explicit, inspectable specifications
- Baseline behavior is reviewed and validated by engineers and domain experts
- System and data dependencies are mapped
- Automated tests are generated alongside analysis, not after delivery
- Risks are addressed and agreed upon up front, not deferred to late-stage testing

03 AI is governed, not autonomous

In these pilots:

- AI accelerates analysis, specification and test generation while understanding how the enterprise works as a whole
- AI outputs are reviewed, validated and approved by domain experts before proceeding
- No behavior change is allowed without a clear evidence trail
- AI increases visibility and consistency for business leaders, while technology leaders maintain ownership of deliverable quality

04 Evidence is produced continuously

As the pilot progresses, teams generate:

- Code-to-spec traceability
- Automated regression tied to original behavior
- Audit-ready artifacts are created as part of delivery, not reconstructed later
- Clear decision points to proceed, pause or stop

This allows risk, compliance and audit teams to engage early with evidence

05 Success is defined by confidence, not speed

A successful pilot of AI-powered modernization delivers three things for the technical side and the business side:

- Reduced uncertainty around system behavior
- Confidence that AI-powered modernization methods can scale safely and/or clarity to stop without sunk cost
- A repeatable, auditable workflow that can be applied to additional systems

Before and after

How Sapient Slingshot systematically reduces risk

The high-level outcomes from modernizing with Sapient Slingshot vary across projects, from millions in projected ROI due to faster technology upgrades, to seven years of an engineer's time saved through mapping thousands of system and data dependencies. But in each scenario, this is exactly why and how utilizing Slingshot achieves safer, faster and better outcomes long-term:

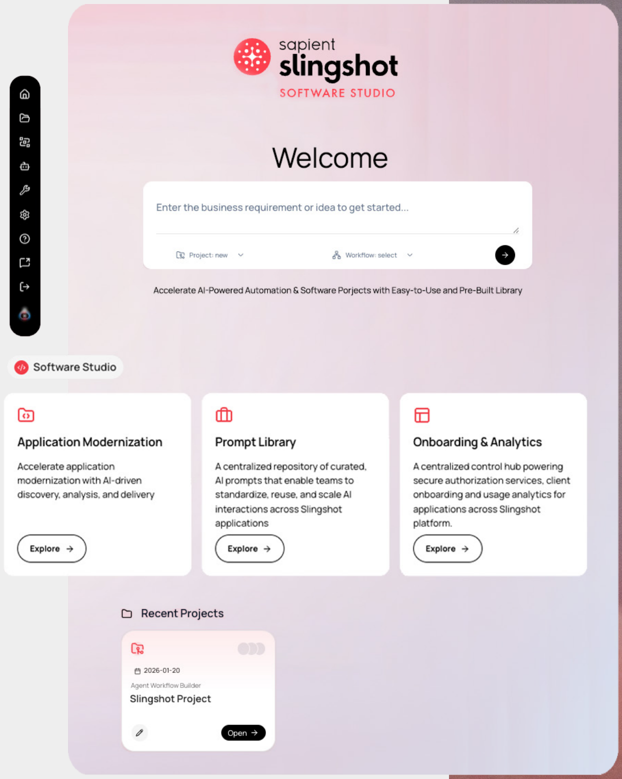
What actually matters	Traditional modernization	AI-enabled approach
How business rules are handled	Old behavior inferred from code and tribal knowledge, rules are re-implemented and hope is placed in testing.	AI analyzes real-world production behavior and code to create verified specifications that prove how existing business rules work.
When risk is discovered	Risk surfaces during integration, audit or production incidents.	AI identifies risks at the beginning by mapping existing business rules to legacy estates and proposed upgrades, using an enterprise context graph.
How compliance proof is created	Compliance is checked near release, creating delays, rework and executive escalation.	AI automatically generates a "paper trail" of evidence for automatic security checks, making compliance efficient and simple.
Speed vs. safety trade-off	You might move fast early, but discover problems later, taking longer to fix them.	AI starts slow, to generate any implied logic behind the existing system, reducing the potential of hidden problems and delays later.
Understanding of system dependencies	Dependencies are discovered through defects, outages or regulatory findings.	Automates mapping of current system dependencies during discovery, so the modernization timeline and approach is efficient and accurate.

The decision facing technology leaders in regulated industries

These real stories prove that modernization will not get safer by waiting. In regulated industries, modernization only gets safer if it becomes more observable, more testable and more governed. The organizations profiled in this playbook did not have to accept more risk to move forward. They engineered risk down through automation, and speed naturally followed.

About Sapient Slingshot

Sapient Slingshot is the only legacy modernization platform that automates the entire software lifecycle end-to-end. Unlike point AI coding assistants, Slingshot pairs a persistent enterprise context graph with specialized SDLC agents to modernize and deliver accurate, fast and governed software. This enables organizations to achieve up to 50 percent savings in modernization cost, 99 percent code-to-spec accuracy and 40 percent productivity gains in new software delivery.



Accelerate AI-Powered Automation & Software Porjects with Easy-to-Use and Pre-Built Library

Application Modernization

Accelerate application modernization with AI-driven discovery, analysis, and delivery

Prompt Library

A centralized repository of curated, AI prompts that enable teams to standardize, reuse, and scale AI interactions across Slingshot applications

Onboarding & Analytics

A centralized control hub powering secure authorization services, client onboarding and usage analytics for applications across Slingshot platform.

To learn more about Sapient Slingshot, the platform used to accelerate legacy modernization across industries, visit: https://www.publicissapient.com/platforms/slingshot