Breaking the 33% Ceiling

How Banking Enterprises Can Industrialise Test Automation and Scale to 60%+ Coverage

A technical briefing for Quality Engineering leaders in UK banking

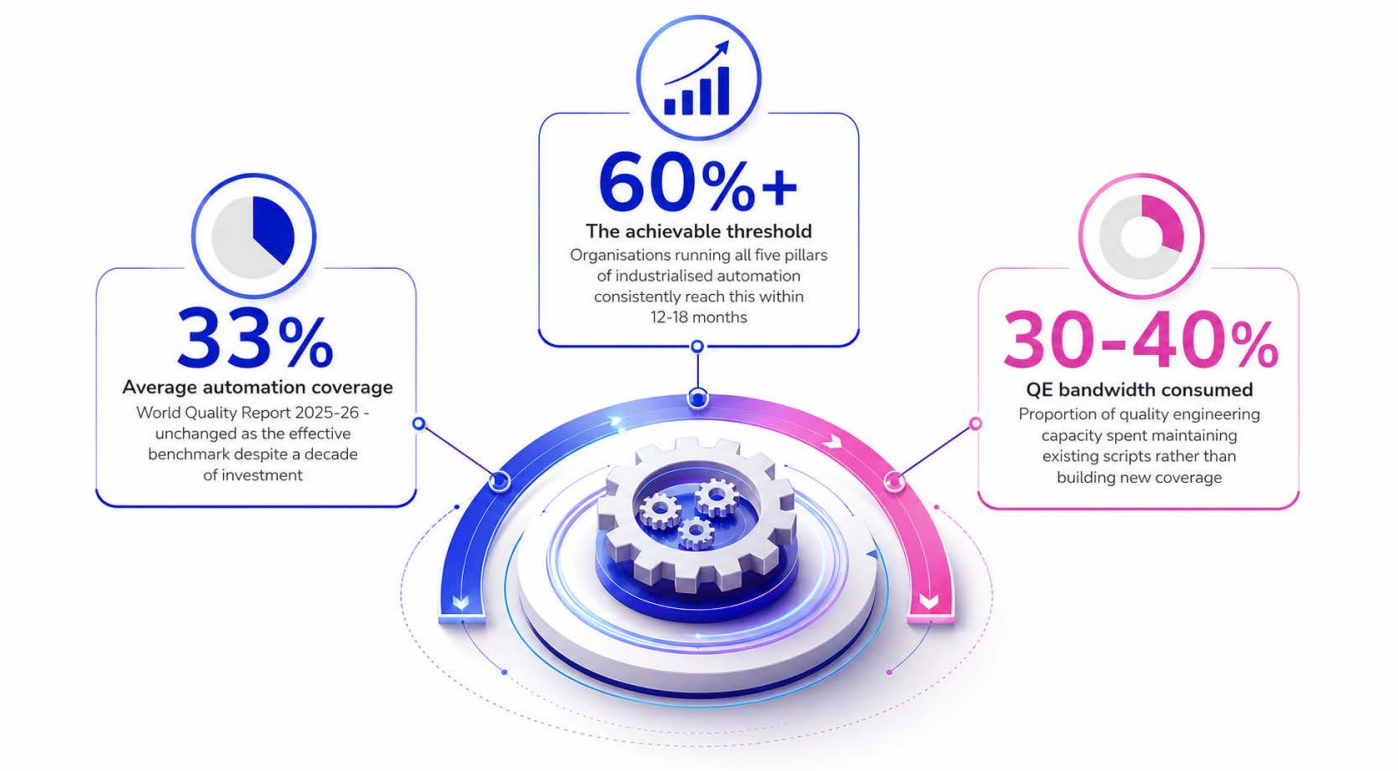
Abstract

Banks have invested in test automation for over a decade. The World Quality Report 2025-26 reveals that average coverage still sits at 33% - meaning two-thirds of all tests run manually on every release cycle. This is not a tooling problem. It is an operating model problem. This paper examines why automation keeps stalling in banking environments, why incremental interventions have not resolved it, and how a five-pillar industrialisation model produces a demonstrable path to 60%+ coverage within 12-18 months of engagement. The argument is grounded in Maveric's delivery experience across UK Tier 2 banks, building societies, and regional institutions running Temenos, Finastra, and legacy core banking systems.

The Automation Plateau - A Problem No One Is Fully Accounting For

Banks have been investing in test automation since the early 2010s. The tools have proliferated, the frameworks have matured, and the intent has never been in question. Yet the World Quality Report 2025-26 - the most authoritative annual benchmark on quality engineering globally, covering organisations across 32 countries - puts average automation coverage at 33%.

That number requires unpacking. It does not mean that 33% of test cases have been automated and the rest are in the queue. It means that in practice, on real release cycles, in live banking environments, two-thirds of the testing burden is still carried manually.



Regression cycles run for weeks. Release confidence depends on human effort. And every time the underlying application changes - which in a modern banking environment happens continuously - some portion of that 33% breaks and needs to be rebuilt.

The cost of this plateau is not merely the inconvenience of manual testing. It manifests in release cycle compression - the inability to deploy with the frequency that digital-native competitors achieve. It manifests in defect escape rates - the proportion of defects that reach production because manual testing cannot achieve the coverage depth that automated suites can. And it manifests in talent attrition: quality engineers who joined to work on interesting automation problems find themselves spending the majority of their time fixing broken scripts.

The question this paper addresses is not whether to scale automation. That argument has been won. The question is why scaling has proved so difficult, and what a delivery model that actually produces scale looks like in practice.

Seven Root Causes - Why Automation Keeps Stalling

Most automation programmes in banking have attempted to fix coverage through incremental intervention: a new tool, a larger scripting team, a framework migration. These interventions have not produced sustained coverage growth because they address symptoms rather than causes. The seven root causes below compound each other - which is why addressing one in isolation rarely produces lasting improvement.

1. Fragmented Operating Model

Automation in most banks is organised by project, not by enterprise function. A payments modernisation programme builds an automation suite for payments. A digital channels programme builds another for mobile. A core banking upgrade builds a third for account management. None of these suites are designed to interoperate. None survive the end of the programme that created them. Coverage resets to zero with every new engagement.

The result is an automation estate that looks substantial on paper - hundreds of scripts, multiple frameworks, significant prior investment - but delivers almost nothing at enterprise scale. When the next release touches multiple systems, the manual effort required to bridge the gaps is enormous.

2. Talent Concentration Risk

Automation in banking has a skills paradox: the engineers who are genuinely capable of building and maintaining enterprise-grade automation frameworks are rare, expensive, and highly mobile. Most QE teams have one or two individuals who are the effective custodians of the entire automation estate. When they leave - and they do leave, regularly - the estate degrades rapidly.

The dependency on specialist tool knowledge compounds this. An automation suite built on a specific version of a specific framework, by a specific engineer who has left, is functionally unmaintainable by the rest of the team. This is not a hypothetical risk. It is the operating reality in the majority of the Tier 2 and regional banks Maveric has engaged with.

3. Test Data Dysfunction

Automation scripts that depend on specific data states fail when those states are unavailable. In banking environments, test data is structurally constrained: production data cannot be used (GDPR, PCI DSS), synthetic data generation is underfunded, and shared test environments are perpetually in contention between multiple delivery programmes.

The World Quality Report 2025-26 reports that 60% of organisations struggle with secure, scalable test data. In banking, the figure is higher. The consequence is automation suites that fail silently - not because the code is wrong, but because the data they depend on is unavailable. Teams stop trusting the suite. Manual testing fills the gap. Coverage decays

4. Wrong Automation Candidates

A consistent pattern in banking automation programmes is the prioritisation of UI-level tests over API and business-process tests. UI tests are visible, demonstrate well in stakeholder reviews, and are relatively straightforward to script against modern frameworks. They are also the most expensive to maintain: they break with every UI change, require constant script updates, and provide coverage of the surface rather than the logic.

The highest-value automation candidates in banking are API-level tests of core business processes: payment processing, account origination, KYC validation, trade settlement. These are stable, high-value, and fast to execute. They are also less visible and less satisfying to demonstrate to stakeholders - which is why they are systematically under-automated

5. Script Brittleness and Maintenance Debt

A brittle automation suite is one where scripts fail for reasons unrelated to application defects. A UI element changes its identifier. An environment configuration differs between test runs. A timing dependency between services causes intermittent failures. These are not test failures - they are maintenance events. But they consume the same engineering capacity as genuine defect identification.

Maveric's delivery experience across UK banking engagements suggests that brittle, high-maintenance scripts consume 30-40% of QE bandwidth across the average automation programme. That is capacity that cannot be deployed on expanding coverage. The maintenance debt compounds: the larger the suite, the more maintenance it requires, the less capacity remains for growth

6. Automation Isolated from CI/CD

Test automation that does not run continuously provides feedback too late. If an automated regression suite runs nightly, a defect introduced at 9am is not detected until the following morning. By that point, the engineer who introduced the change has moved on to the next feature. Context is lost. Fix time is longer.

In banking environments, CI/CD adoption has accelerated significantly in the past three years - particularly in challenger banks and the digital channels of Tier 2 institutions. But the automation estate has not kept pace. Suites designed to run as batch processes are incompatible with continuous delivery pipelines. The result is automation that cannot gate releases - which means it cannot fulfil its fundamental purpose.

7. Legacy Platform Exclusion

The most consequential gap in most banking automation estates is the one that is never discussed in programme reviews: core banking systems are almost entirely outside the automation perimeter. Temenos T24/Transact, Finastra Fusion, Oracle FLEXCUBE - the systems that process every significant transaction in the bank - have testing estates that are predominantly manual.

The reasons are well understood: complex APIs, batch-oriented architectures, environment constraints, and the genuine difficulty of building automation against systems with limited API exposure and undocumented integration behaviour. But the consequence is structural: a bank can automate 80% of its digital channel tests and still only reach 40% of total coverage if the core banking layer is excluded. The ceiling is architectural, not motivational.

Why Incremental Fixes Have Not Worked

The banking industry has applied the following interventions, in roughly this order, over the past decade. None have produced sustained coverage growth at enterprise scale. Understanding why they have failed is essential context for what actually works.

Intervention	Why It Does Not Resolve the Root Causes
Buying more tools	Tool proliferation without a unified operating model adds complexity, not coverage. The 33% ceiling is not caused by tooling gaps - it is caused by the seven structural factors above. A new tool is adopted by the same fragmented programme structure, built by the same talent concentration, and deployed against the same wrong automation candidates. Coverage does not grow.
Adding headcount	Hiring more automation engineers without changing the delivery model scales cost, not automation. New engineers join existing teams, inherit existing brittle suites, and spend the majority of their time on maintenance. The maintenance-to-build ratio does not improve because the root causes of maintenance remain unaddressed.
Framework migrations	Switching from Selenium to Playwright, from Playwright to Cypress, from Cypress to the next generation - restarts the clock without addressing root causes. The new framework inherits the same operating model: project-level ownership, wrong automation candidates, no CI/CD integration. Coverage resets. Maintenance debt rebuilds.
AI tool pilots	Experimenting with AI-powered testing tools - test generation, self-healing, intelligent test selection - without an integration strategy produces compelling demos and limited production-grade coverage. The tools are genuinely capable. But deployed into a fragmented operating model with no reusable asset base and no pipeline integration, they optimise at the script level rather than the enterprise level.
Centralising the framework	Building a 'centre of excellence' or shared automation framework without the supporting governance, reusable asset strategy, and CI/CD integration creates a framework that teams are told to use and do not. Without business-process-aligned assets that development teams can contribute to and consume, centralisation produces a standards document, not a coverage model.
Buying more tools	Tool proliferation without a unified operating model adds complexity, not coverage. The 33% ceiling is not caused by tooling gaps - it is caused by the seven structural factors above. A new tool is adopted by the same fragmented programme structure, built by the same talent concentration, and deployed against the same wrong automation candidates. Coverage does not grow.

The Five-Pillar Model | Industrialising Automation

The organisations that have broken through the 33% ceiling have one characteristic in common: they stopped treating automation as a downstream QA activity and started treating it as an engineering discipline embedded across the software delivery lifecycle. This is not a technology change. It is an operating model change - and it requires five capabilities operating together.

Pillar 1: Embedded Automation

Automation is not a phase that follows development. It is integrated into sprint planning, definition of ready, and release gate criteria from the first day of every feature cycle. Engineers and testers co-own automation from requirement to deployment. Defects found in automated execution during development are fixed in the same sprint - not passed to a QA phase. The shift-left principle, applied rigorously, compounds over time: coverage grows with every sprint rather than being rebuilt from zero at the start of each regression phase.

In practice, this means automation stories sit in every product backlog alongside feature stories. It means automation engineers are embedded in delivery teams rather than organised in a separate QE function. And it means the definition of done for any feature includes passing automated test coverage, not just manual sign-off

Pillar 2: Reusable, Process-Aligned Assets

The fundamental economic problem with script-by-script automation is that the investment does not compound. Each script is built for a specific version of a specific application at a specific point in time. When the application changes, the script breaks. The investment is lost.

Business-process-driven test libraries change this dynamic. A reusable automation component for a payment initiation process - built to abstract away the specific UI implementation - survives UI changes because it operates at the business logic level. A KYC validation suite built against the API layer remains valid across multiple front-end iterations. A trade settlement regression library, once built and governed, can be consumed by any programme that touches trade processing - rather than being rebuilt by each programme from scratch.

Maveric's delivery approach centres on building these libraries first, in the domains where the bank's coverage gap is most costly: core banking transactions, payments processing, KYC and onboarding, regulatory reporting. Coverage compounds over programmes rather than resetting

Pillar 3: Low-Code Accelerators

The talent concentration problem - the dependence on a small number of specialist automation engineers - cannot be solved by hiring. There are not enough automation engineers with banking domain knowledge to staff every delivery programme. The solution is democratisation: making automation contribution accessible to functional testers and subject matter experts who understand the business processes but cannot write test code.

Low-code automation interfaces allow functional testing professionals to build and execute automated test cases without scripting expertise. The technical automation layer remains the responsibility of automation engineers - but the coverage expansion layer can be driven by the people who understand the business process best. This is how coverage scales beyond the capacity of the automation engineering team.

Pillar 4: AI-Enabled Maintenance and Scale

Script brittleness consumes 30-40% of QE bandwidth. AI-assisted automation addresses this at the maintenance layer - not as a replacement for engineering judgement, but as an accelerator that reduces the manual overhead of keeping the suite current.

Self-healing test scripts adapt to UI and API changes automatically, reducing the maintenance event volume by a significant fraction. AI-assisted test generation reduces the time from requirement specification to automated test case, shortening the coverage lag that opens every time a new feature is deployed. Predictive test selection - running the subset of the automation suite most likely to catch defects introduced by a specific code change - reduces execution time and improves feedback latency.

Applied pragmatically, within a coherent operating model, AI-assisted automation produces sustainable scale. Applied as a standalone tool without the supporting pillars, it produces demos.

Pillar 5: Pipeline-Native Continuous Execution

Automation that does not run continuously cannot gate releases. The final pillar is architectural: the automation suite is integrated into the CI/CD pipeline, executes on every build, and produces coverage metrics and quality signals in real time. Release decisions are informed by automated evidence, not by the outcome of a manual testing phase.

In banking environments, this requires deliberate integration work - particularly for the core banking and legacy system layers that have historically sat outside the pipeline. Maveric's approach includes API-layer automation strategies for systems that do not support direct UI automation, environment provisioning approaches that allow stable test data to be available on every pipeline run, and execution architecture that keeps suite run times within the feedback window that continuous delivery requires.

What Changes When Coverage Doubles - The Business Case

The business case for moving from 33% to 60%+ coverage is not primarily a quality argument. It is a delivery economics argument. The table below maps the operational reality at 33% against the operational reality at 60%+, across the dimensions that matter to delivery leaders in banking.

Dimension	At 33% Coverage	At 60%+ Coverage
Regression cycle duration	3-6 weeks for full regression. Release cadence constrained to monthly or longer.	5-10 days or less. Fortnightly or continuous releases achievable.
Defect escape rate	Higher - manual testing cannot achieve the coverage depth or consistency of automated suites across complex integration chains.	Materially lower - automated suites execute consistently across all covered paths on every build.
Manual testing effort	60-70% of test execution is manual. QE team capacity dominated by execution, not coverage building.	35-40% manual. QE team capacity available for exploratory testing, edge cases, and coverage expansion.
Script maintenance burden	30-40% of QE bandwidth consumed by maintaining brittle scripts. Net new coverage capacity severely constrained.	10-15% - self-healing and process-aligned libraries reduce maintenance events significantly.
Response to application change	Each significant application change triggers a maintenance cycle before regression can run. Coverage degrades on every release.	Pipeline-native execution detects failures immediately. Self-healing reduces maintenance cycle time.
Legacy system coverage	Core banking and integration layers predominantly manual. The highest-risk test scenarios have the least automated coverage.	API-layer automation strategies bring core banking, payments, and regulatory reporting into the automated perimeter.
Talent leverage	Automation engineers occupied with maintenance. Domain knowledge of functional testers unused in automation.	Low-code accelerators allow functional testers to contribute to coverage. Automation engineers focused on architecture.
Release confidence	Depends on the judgement and experience of the manual testing team. Variable. Difficult to demonstrate to regulators.	Evidence-based. Automated coverage metrics provide an auditable, consistent basis for release decisions.

What Changes When Coverage Doubles - The Business Case

The five pillars above describe a delivery model. What they do not fully capture is the degree to which banking domain expertise determines whether that model produces results. Generic test automation frameworks - built for e-commerce, SaaS, or enterprise software - do not understand that a payment instruction has a correspondent bank chain with SWIFT message formatting requirements. They do not understand that a KYC workflow has regulatory timing obligations that differ between jurisdictions. They do not understand that a trade settlement failure has downstream consequences across custody, reporting, and regulatory capital calculation that a generic regression suite will never detect.

Maveric's QE practice is banking-exclusive. The reusable asset libraries we build are organised by banking business process: retail current accounts and lending, payments (SEPA, SWIFT, Faster Payments, Open Banking), KYC and financial crime compliance, corporate lending and trade finance, wealth management and investment operations. These are not generic test cases with banking terminology applied. They are automation components built from deep understanding of how these processes work, what goes wrong, and where the highest-risk test scenarios sit. This is the differentiation that matters for a Tier 2 or regional bank: a QE partner that arrives with pre-built, banking-specific reusable libraries, rather than arriving with a framework and a methodology and requiring the bank to fund the domain knowledge build from scratch

The Starting Point - Automation Coverage Assessment

The path from 33% to 60%+ begins with a clear-eyed assessment of where the current automation estate actually stands - not where the project documentation says it stands.

Automation Coverage Assessment

A structured 2-day engagement. Maveric reviews the bank's current automation estate across five dimensions:

Coverage model: what is actually automated, at what layer, and against which business processes

Maintenance ratio: proportion of QE capacity consumed by script maintenance versus new coverage development

Pipeline integration: degree to which the automation suite is integrated with CI/CD and can gate releases

Domain coverage gaps: which business process layers (core banking, payments, KYC, trade) are outside the automated perimeter

AI and low-code readiness: the fastest path to coverage uplift given the bank's existing tooling and team structure

Deliverable: A prioritised Automation Maturity Gap Report — with a specific, sequenced path to 60%+ coverage and ROI modelling against your current release cycle costs.

Fee: £3–5k fixed.

To arrange an Automation Coverage Assessment or discuss the QE service line connect with us

The question is not whether to scale automation. The question is how long it costs you to keep running at 33%.

Maveric Systems

is a banking-exclusive technology specialist with over 25 years of domain expertise. We partner with global financial institutions to engineer trust in AI-first banking.

While others apply AI at the periphery, we embed it at the core through our proprietary AI @ Scale framework and AI-powered platforms and solutions. Guided by principles that engineer trust, we embed fairness, explainability, reliability, and compliance into every AI solution by design. This enables responsible AI adoption at scale.

Our domain depth across operations and technology in retail and corporate banking, wealth management, and capital markets, combined with a pragmatic, outcome-driven delivery model, ensures that every AI initiative is rooted in contextual relevance and precision.

Backed by dedicated AI Centers of Excellence, a powerful ecosystem of technology platforms, and recognition from leading industry bodies, we are the trusted engineering partner for the AI era.

The Maveric Edge

25+
years of domain mastery

7+
Operations and Technology AI CoEs

12+
AI powered Platforms and Solutions

AI@Scale
Scale Proprietary Framework for AI adoption at Scale

ROHIT BHOSALE
Sales VP - UK & Europe
rohitbhosale@maveric-systems.com
M: +44-7468539190

MAVERIC