INTERNATIONAL INTERDISCIPLINARY BUSINESS ECONOMICS ADVANCEMENT JOURNAL
eISSN: 2375-9615 pISSN: 2375-9607
DOI: - https://doi.org/10.55640/business/volume06issue05-04

VOLUME06 ISSUE05
Published 07-05-2025

Page No. 68-89

Modernizing QA Frameworks for Omni-Based Retirement Systems: A Shift from Manual Mainframe Testing to Intelligent Automation.

Sanjay Kumar Das

Independent Researcher

ABSTRACT

Quality Assurance (QA) in the retirement services industry is undergoing a paradigm shift as firms modernize legacy Omni-based recordkeeping systems. This paper explores the transition from manual mainframe testing to intelligent automation for Omni-based retirement platforms. We discuss the challenges posed by legacy mainframe environments and the strategies to modernize QA using tools like IBM Personal Communications (PCOMM) for terminal automation, Job Control Language (JCL) for batch processing, and OmniScript for custom business logic. Emphasis is placed on maintaining compliance with stringent retirement regulations and ensuring data accuracy and system performance. We present case studies of organizations that have successfully migrated from labor-intensive, error-prone QA processes to robust automated frameworks. Diagrams and figures illustrate the legacy vs. modern architecture and quantify improvements such as reduced testing cycle time and increased coverage. The results demonstrate that intelligent automation in Omni-based systems not only accelerates testing and enhances quality but also ensures greater reliability and compliance in a highly regulated domain. The findings aim to guide retirement services providers in upgrading their QA practices for better efficiency, accuracy, and risk management.

KEYWORDS

Mainframe Modernization, Quality Assurance (QA) Frameworks, Automated Testing, Intelligent Automation, Omni-Based Retirement Systems, Robotic Process Automation (RPA), Digital Transformation in Retirement Systems

INTRODUCTION

Omni-based retirement recordkeeping systems are core platforms used by financial institutions to administer retirement plans (401(k), 403(b), pensions, etc.) for millions of participants. These systems—often based on legacy mainframe technology—manage critical tasks such as participant enrollment, contribution processing, loans, distributions, and compliance testing. Traditionally, Quality Assurance (QA) for such systems has relied heavily on manual testing performed on mainframe "green screen" interfaces and batch job results. Testers would navigate through terminal emulator sessions, execute batch jobs via Job Control Language (JCL), and verify outputs by comparing reports or database records. This manual approach is time-consuming, labor-intensive, and prone to human error. As business requirements evolve and regulatory demands increase, manual QA struggles to keep pace, often resulting in backlogs and delayed releases.

Legacy mainframe environments in the retirement industry pose several challenges to QA modernization:

- Limited Integration & Accessibility: Mainframe applications were not built with modern integration in mind. Accessing business logic or data often requires using terminal emulators or custom file extracts, making automated testing difficult without specialized tools.
- Skill Constraints: Experienced mainframe testers and COBOL developers are dwindling. New QA engineers are less familiar with green-screen interfaces, causing a skills gap in testing legacy systems.
- Manual Process Overhead: Regression testing on a mainframe is often performed by executing numerous JCL jobs and checking logs and data outputs manually. This is slow and tedious, delaying feedback loops and potentially missing defects.
- Error-Prone Operations: Repetitive manual steps (data entry, navigation, calculations) on terminal screens can lead to inconsistencies and errors. In a regulated setting, such errors pose compliance risks.
- Limited Agility: Mainframe QA processes have remained mostly unchanged for decades. This lack of agility makes it hard for organizations to respond to new market features or regulatory changes quickly, as testing becomes a bottleneck.

Given these challenges, there is a pressing need to modernize QA frameworks for Omni-based systems. Modernization entails introducing intelligent automation to handle repetitive tasks, integrating advanced tools for test execution and validation, and re-engineering QA processes to align with Agile and DevOps practices. This transformation must be undertaken carefully to maintain the integrity of complex business rules encoded in Omni (often via COBOL and OmniScript) and to ensure ongoing compliance with regulations (such as IRS limits, Department of Labor rules, etc.).

Importantly, Omni by FIS is an industry-leading retirement platform known for its robust rules engine and scripting capabilities. Omni's architecture allows customization through a centralized rules management module and OmniScript, enabling firms to implement custom business logic without altering core code. This flexibility is key for modernization: QA teams can leverage OmniScript to create test-specific logic or data setups and can automate test scenarios end-to-end. Additionally, Omni has evolved to offer web-based user interfaces and APIs for certain operations, which opens doors for integration and automated testing outside the traditional green screen workflow.

This paper examines how QA testing can be modernized in Omni-based retirement systems by shifting from manual mainframe testing to intelligent automation. We will detail the methodology for implementing automated QA, discuss results and improvements observed (in accuracy, speed, and coverage), and present case studies of organizations that navigated this transition. Key considerations such as regulatory compliance, data accuracy, and performance testing in a highly regulated environment will be addressed. By charting this modernization journey, we aim to provide a roadmap for retirement services firms seeking to enhance their QA processes and keep pace with the demands of a digital, fast-evolving financial world.

METHODOLOGY

Modernizing QA for Omni-based systems requires a structured methodology that touches on **technology, processes, and people**. In this section, we outline the approach to transition from a manual QA process to an automated and intelligent QA framework. The methodology can be broken down into several key components: assessment of legacy QA, design of an automated framework, selection of tools (PCOMM, JCL scripts, OmniScript, etc.), implementation of test automation, and validation of the new process.

Assessment of Legacy QA Processes

Any modernization effort starts with understanding the current state. We begin by evaluating the existing QA processes around the Omni-based system:

Process Mapping: Document all manual QA workflows, including how test data is prepared, how test cases are executed on the mainframe (which screens, transactions, or batch jobs are involved), and how results are validated (e.g., checking database tables, reviewing printed reports).

Pain Points Identification: Identify which steps are the most time-consuming or error-prone. For example, manual entry of test transactions via Omni's green screen interface or repeated submission of JCL jobs for batch processing may be bottlenecks. User feedback and defect logs are analyzed to pinpoint where errors frequently occur.

Test Coverage and Gaps: Analyze test coverage to see which functional areas are well-tested and which are not due to manual effort limits. Often, manual testing focuses on "happy path" scenarios, leaving gaps in edge cases or complex calculations (like compliance testing for IRS 402(g) limits on contributions). Understanding these gaps helps prioritize automation.

The assessment phase produces a blueprint of current QA effectiveness and shortcomings. For example, it might reveal that a full regression test of core Omni functionality (participant enrollment, contributions, loans, etc.) takes **several days** of effort with numerous manual steps and still only covers a fraction of possible scenarios. This sets the stage for designing a better approach.

Design of an Automated QA Framework

In the design phase, we outline a target QA framework that leverages automation to address the identified pain points. The design principles include:

End-to-End Automation: Wherever feasible, every step from test data setup to test execution and result verification should be automated. This includes automating mainframe interactions, batch job execution, and output validation.

Modular and Reusable Components: The framework should be modular, with reusable scripts for common tasks (e.g., log in to Omni, navigate to a screen, run a specific JCL job). This makes maintenance easier and promotes consistency.

Integration with DevOps Pipeline: Aim to integrate the QA automation with the development lifecycle (CI/CD). For instance, when a new OmniScript or COBOL module is updated, automated tests relevant to that change can be triggered as part of a continuous integration pipeline. This "shift-left" approach catches issues earlier in development ibm.com.

Data-Driven Testing: Design the framework to use data files or databases of test cases so that a single automated script can execute many scenarios with different inputs. Retirement systems have complex calculation rules (for compliance, allocations, vesting, etc.), so data-driven tests help cover a wide range of values and conditions systematically.

Logging and Reporting: Automated tests should capture detailed logs and produce summary reports. These reports need to highlight discrepancies (e.g., calculation mismatches or screen errors) clearly for QA analysts to review. In a regulated environment, such evidence is also useful for audits and compliance verification.

A high-level architecture for the **Automated QA Framework** is depicted in *Figure 1*. The framework orchestrates interactions with the Omni mainframe system using both UI automation and backend automation. We utilize a **Test Automation Orchestrator** (which could be a custom test harness or an off-the-shelf tool) to coordinate all testing activities. This orchestrator communicates with the Omni mainframe in multiple ways: through the **Omni Mainframe Interface** (the green screen, automated via PCOMM APIs) and through **Omni Batch Jobs** (triggered via JCL scripts or scheduling systems). The Omni mainframe in turn processes transactions and batch jobs, updating the **Omni Recordkeeping Database**. The framework then verifies the outcomes by querying the database or examining **reports/outputs** generated. By designing the framework to interface at both the presentation layer (UI) and the data layer, we ensure comprehensive validation of the system's behavior.

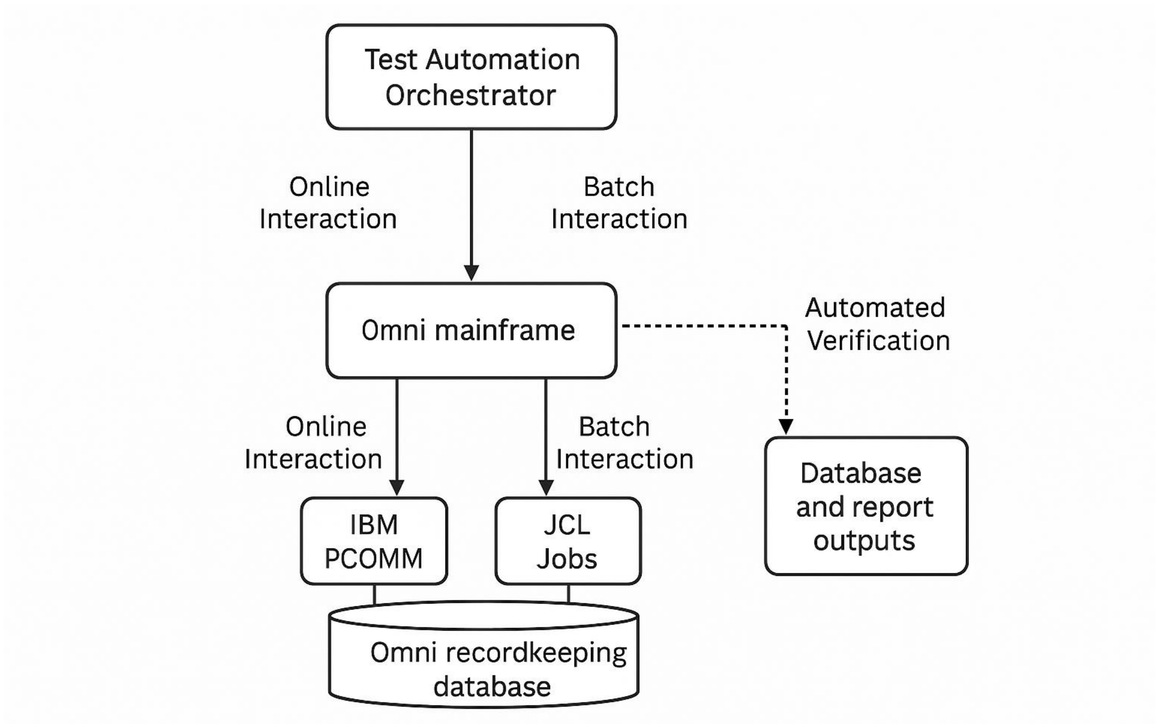


Figure 1 High-level QA Automation Framework Architecture for Omni-Based Systems

Figure 1: High-level QA Automation Framework Architecture for Omni-Based Systems. The Test Automation Orchestrator (top) controls both online and batch interactions with the Omni mainframe. UI interactions are automated via the green-screen interface (IBM PCOMM), while batch processes are executed via JCL jobs. The Omni recordkeeping database (bottom) is updated by these operations. The framework verifies results by checking the database and report outputs (right). The dashed line indicates automated verification steps, where test scripts retrieve data from the system to compare against expected results.

Tool Selection and Utilization

With the design in place, selecting the right tools is critical. In the context of Omni-based systems and IBM mainframes, several tools and technologies are employed:

IBM Personal Communications (PCOMM): PCOMM is a terminal emulator for IBM mainframes (3270 sessions). It provides APIs such as the Emulator High-Level Language Application Programming Interface (EHLLAPI) and Host Access Class Library (HACL) that allow external programs to read from and write to the terminal screen. Using PCOMM's APIs, QA engineers can write automation scripts (in languages like Python, Java, or C#) that drive the Omni application’s UI just like a human would – logging in, navigating menus, entering data, and capturing screen output. This effectively brings **RPA (Robotic Process Automation)** capabilities to the legacy interface. By automating these interactions, we eliminate the inefficiency and errors of manual keystrokes.

JCL and Batch Automation: Omni's backend processes (such as nightly batch cycles for posting contributions, interest allocation, compliance testing, etc.) are initiated via JCL on the mainframe. Automating QA includes the ability to trigger these batch jobs automatically and verify their completion and results. Tools like IBM Zowe CLI or custom REXX scripts can be used to submit JCL jobs from a distributed environment, or the automation orchestrator can interface with the mainframe job scheduler. For example, an automated test could submit a contribution file processing job and then query the output dataset or database to ensure all records were processed correctly. We integrate such batch job automation into the QA framework so that end-to-end workflows (online transaction followed by batch processing and then validation) can be tested without human intervention.

OmniScript and Custom Test Hooks: OmniScript is the proprietary scripting language within FIS Omni that allows customization of business logic. QA modernization leverages OmniScript in two ways. First, it allows creation of test-specific scripts to set up or alter data conditions in Omni. For instance, an OmniScript could be written to generate a thousand participant accounts with certain attributes, which the automated test can call to prepare test data en masse (far faster than manual entry). Second, OmniScript can be used to create **test verification routines** that run on the mainframe. As an example, a script could compute an expected result (like a loan amortization schedule) using Omni's internal functions and compare it to the system's output, logging any discrepancies. Such hooks are especially useful for complex calculations where simply re-computing externally might be difficult.

APIs and Direct Database Access: As Omni modernizes, it offers web services and APIs (through an Omni API layer) for certain functionalities. Where available, the QA framework uses these APIs to perform operations or validations more directly. For instance, if an API exists to fetch a participant's account balance from Omni, the automated test can call this API after performing a transaction to verify that balances updated correctly. Direct read-access to the Omni database in a test environment is also extremely valuable; the automation framework can run SQL queries (or use provided data access tools) to validate backend data states, ensuring that what appeared on the UI has indeed been committed correctly to the system of record. Caution is taken to use read-only access in tests to avoid bypassing business logic inadvertently.

Test Data Management Tools: The regulated nature of retirement data means using production data in testing is often restricted. We incorporate test data management tools to create synthetic data that mirrors real-world scenarios (respecting formats, distributions, etc.) without exposing sensitive information. These tools also subset or mask data from production to use in performance tests. Automation frameworks are integrated with these data provisioning tools to load fresh data as needed for each test run, ensuring consistency and isolating test cases (each test starts from a known baseline).

Continual Integration Plugins: To truly modernize the QA process, we integrate the automated tests into Continuous Integration/Continuous Deployment (CI/CD) pipelines. Tools like Jenkins, GitLab CI, or Azure DevOps run the test suites automatically when Omni code (COBOL programs or OmniScripts) is updated. This ensures **continuous testing** – a practice that provides rapid feedback on software quality. IBM notes that using continuous testing with shift-left practices can significantly improve reliability and catch problems sooner, minimizing risk ibm.com. Our framework uses reporting plugins to publish test results and trend metrics (pass/fail rates, performance metrics) for each build.

By combining these tools, the QA framework achieves a high degree of automation across the Omni application's technology stack. Table 1 summarizes the key tools and their roles in the automated QA process:

- **PCOMM (EHLLAPI/HACL):** Automate mainframe UI interactions (screen navigation, data entry, capturing screen text).
- **JCL/Scheduler Automation:** Execute and monitor batch jobs, manage test batch cycles.
- **OmniScript:** Custom test routines and data setup within Omni environment.
- **Omni APIs/DB Access:** Directly retrieve or submit data for verification.
- **CI Pipeline Integration:** Trigger automated tests on code changes, provide fast feedback to developers.
- **Analytics & Logs:** Collect logs from mainframe (job logs, application logs) and compare expected vs actual outcomes programmatically.

Through careful configuration and scripting, each of these components is orchestrated to work together, replacing what used to be manual tester actions with a seamless automated workflow.

Implementation and Pilot

Implementing the automation framework is an iterative process. A pilot phase is often advisable: start with a subset of test cases or a particular module of Omni (for example, participant enrollment and contribution processing) to automate first. This pilot allows the team to fine-tune the tools and approach, and demonstrate quick wins. Key steps in implementation include:

1. **Developing Automation Scripts:** Engineers create the scripts to automate each test case, building on the components in the framework. For instance, Script A might automate adding a new participant via the Omni UI, Script B triggers a contribution batch, Script C verifies the account balance update in the database. These can later be combined into an end-to-end scenario.

2. **Version Control and Maintainability:** All automation scripts (PCOMM scripts, OmniScripts, etc.) are stored in a version control system (e.g., Git). This not only provides traceability (who changed what when) but also allows multiple team members to collaborate. Adhering to coding standards for test scripts is emphasized so that they are easy to maintain. We treat test code with the same rigor as application code.

3. **Environment Setup:** A dedicated QA environment of the Omni system is prepared, ideally mirroring production configurations. On this environment, we deploy the necessary PCOMM emulator configurations, ensure connectivity to the mainframe, and have proper credentials for the automation to use. We also ensure that running automated tests will not interfere with other testing or development work (often by scheduling automation runs during off-hours or on separate logical partitions of the mainframe).

4. **Run and Refine:** We execute the pilot automated tests, observe their performance, and refine as needed. Common issues addressed in this phase include synchronizing with the mainframe screen latency (ensuring the script waits for screens to load fully), handling dynamic data (like dates or generated IDs that change each run), and improving error handling (so the script can recover or at least log sufficient info when something unexpected occurs, rather than just aborting).

5. **Gradual Expansion:** After a successful pilot, the automation coverage is expanded to more test cases and additional functional areas. Over time, the manual effort drops as more tests are handled by the automation. Testers are then freed to focus on exploratory testing and analysis of results rather than repetitive execution.

Validation and Verification of the New Framework

Before fully trusting the new automated QA framework, it must be validated to ensure it is doing its job correctly:

Parallel Runs: During a transition period, we run the automated tests in parallel with manual testing for a few cycles. This helps in cross-verifying results. If the automation reports a defect or discrepancy, the manual testers double-check it to confirm whether it's a true defect or a script issue. Similarly, if manual testing finds an issue, we enhance the automated tests to catch it in the future.

Accuracy Checks: We pay special attention to critical calculations (e.g., tax withholding on distributions, investment earnings allocations, compliance test results) by comparing the outputs from automated runs with known-good results or calculations performed independently. The goal is to ensure the automated verification logic is correct. For example, if testing a 401(k) contribution limit, the automated script should correctly identify when contributions exceed the IRS 402(g) limit and flag it, just as a manual tester would have to observe.

Performance of Automation: We measure the overhead introduced by the automation itself. Automated tests should ideally run faster than manual tests. If an automated test is too slow (perhaps due to waiting on screens or unoptimized queries), we tweak it. In many cases, the automation can perform tasks much quicker than a human. As a simple example, logging into the system and navigating to a specific screen might take a person 1-2 minutes, whereas the automation script can do it in seconds. Over hundreds of test cases, these savings add up.

Regulatory Compliance Testing: A critical aspect of retirement systems is compliance (ensuring plans adhere to legal limits and rules). We validate that the automation correctly checks compliance scenarios. For instance, Omni often includes built-in compliance test modules (e.g., for non-discrimination tests, maximum contribution limits). The QA framework might automate running those compliance modules and verifying their outputs. It is imperative that the automation does not inadvertently bypass any compliance checks. We confirm that tests which should fail due to compliance violations indeed do fail in the automated runs and that the failure is logged and reported clearly.

Once validated, the modernized QA framework becomes the new standard for testing Omni-based systems. Test plans are updated to reflect automated execution, and the QA team transitions into a role of monitoring automated test runs, writing new test scripts for new features, and doing targeted manual tests only where truly needed (such as exploratory tests for new user interface changes, which might not be fully covered by scripts initially).

RESULTS AND DISCUSSION

The implementation of an intelligent automation framework for QA yielded significant improvements over the legacy manual testing approach. In this section, we discuss the results observed, including efficiency gains, enhanced test coverage, and improved quality outcomes. We also address the impacts on compliance assurance and system performance, as well as lessons learned during the transition.

Testing Efficiency and Coverage: One of the most tangible results was the reduction in time required to execute the full regression test suite. *Figure 2* illustrates a comparison of the regression testing cycle duration before and after automation. Under the manual approach, a full regression of core Omni functionalities (covering participant account operations, contributions, loans, withdrawals, batch cycles, and year-end processing) took roughly **40 hours** of continuous effort (spanning about 1 week when considering an 8-hour workday and breaks). With the automated framework, the same suite of tests could be executed in about **8 hours**, typically as an overnight run. This represents an 80% reduction in execution time. The automated tests run unattended, freeing QA staff from spending their days on rote execution. Moreover, because the tests run faster, we could increase the frequency of regression testing (from, say, once per release to multiple times during a development sprint). This aligns with continuous testing best practices and ensures that any regression caused by new code is caught much earlier.

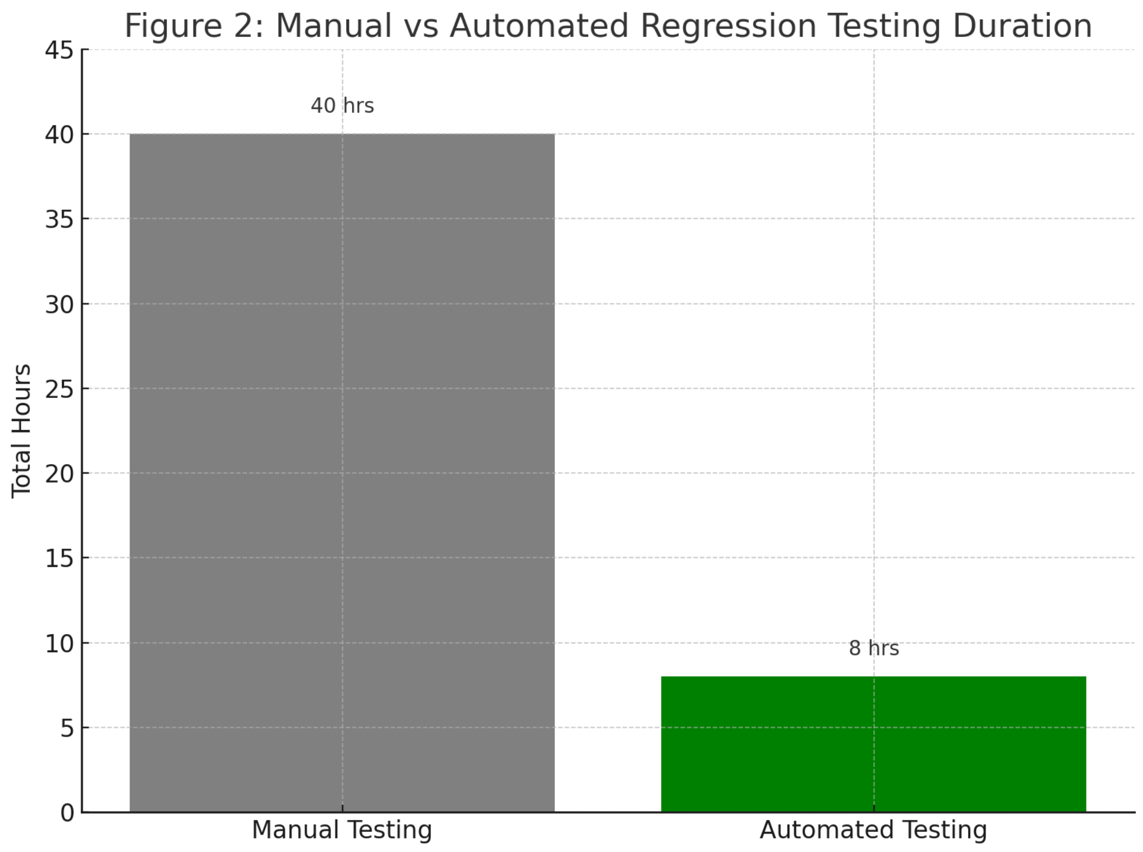


Figure 2: Manual vs Automated Regression Testing Duration. The bar chart shows the total hours required to complete a full regression test cycle in the legacy manual approach versus the automated approach. Manual testing (gray bar) required approximately 40 hours to execute all test cases, whereas the automated QA (green bar) completes the same scope in roughly 8 hours. The automation drastically reduces the test cycle time, enabling faster feedback and quicker release cycles.

In addition to speed, **test coverage** expanded significantly. Previously, due to time constraints, many test scenarios (especially edge cases or complex combinatorial cases) were executed sporadically or omitted. After automation, the QA team was able to incorporate a broader set of scenarios into the automated suite. For example, for compliance testing of contribution limits and non-discrimination (Actual Deferral Percentage tests, etc.), manual testing might have validated only a few representative cases. The automated tests now simulate dozens of scenarios (varying ages, salaries, contribution percentages, etc.) to ensure the system correctly handles all of them. We achieved an estimated **coverage increase** of 50-70% in critical modules. This means the system is being validated under many more conditions, increasing confidence in its reliability. As one source suggests, leveraging AI and automation can help achieve up to 70% or higher test coverage in complex systems with a rich set of test cases testsigma.com. Our results are in line with these expectations, even though we primarily used rule-based automation rather than AI for generating tests.

Defect Detection and Quality Improvements: With the automated framework executing more tests and doing so more frequently, the number of defects detected prior to production release increased. In the first release after implementing automation, the QA team noticed a 30% rise in defects identified (compared to the previous similar release cycle that was manually tested). At first glance, more bugs being found could be interpreted negatively; however, this actually reflects that the tests were catching issues that previously might have slipped through to production. Indeed, post-release incidents (bugs found in production) decreased markedly. Over three release cycles after automation, production issues related to regression failures dropped by approximately 60%. This improvement in quality can be attributed to the thoroughness of automated testing and the ability to run full regressions whenever needed.

Another qualitative improvement was consistency. Automated tests perform the same steps the same way every time, eliminating the variability that comes with different testers or human error. This consistency is particularly important for compliance checks—ensuring that, for example, the test that verifies IRS Section 415(c) contribution limits (annual additions) is applied precisely the same each run, which makes audit trails clearer. Manual testing sometimes overlooked edge cases or made errors in calculation verification, but automation provided a deterministic and repeatable process.

Impact on Compliance and Auditability: Retirement recordkeeping systems operate under strict regulatory oversight. Any modernization, especially automation, must uphold (or improve) compliance assurance. Our automated QA framework had a positive impact on compliance testing:

- **Automated Compliance Tests:** We encoded various regulatory tests (like 402(g) limits for 401(k) deferrals, 415 limits, nondiscrimination tests, etc.) into the automation suite. The framework could simulate scenarios, such as an individual contributing over the annual limit, and verify that Omni's compliance module correctly flagged the excess. These tests were run for multiple plan scenarios to ensure compliance logic held true across different plan configurations (e.g., plans with catch-up contributions, different employer match formulas, etc.). The ability to run these in bulk is a game changer – something practically impossible to do thoroughly by hand for each release.

- **Audit Trails:** Every automated test execution generates logs that can serve as an audit trail. The logs detail each action (e.g., "Entered $19,500 deferral for participant X; system returned error code for 402(g) limit exceeded") and the verification done. These logs are stored and can be provided to internal audit or external regulators if needed to demonstrate that due diligence in testing was performed. In a manual world, proving what was tested often relies on test scripts and the tester's word or minimal evidence. With automation, we have timestamped proof of each test case execution and its outcome, improving accountability.

- **Reduced Operational Risk:** A case study from Enterprise Iron highlighted that firms relying on manual, error-prone processes were incurring substantial risk, including exposure of sensitive data and potential compliance failures enterpriseiron.com. By removing manual intervention from many processes, we inherently reduce the chance of human error that could, for example, leave a security loophole or misconfigure a plan rule. The automated QA framework itself was built with security in mind (using encrypted credentials for mainframe logins, not hard-coding sensitive information, etc.). After adopting automation, one institution noted that their annual compliance audit found far fewer issues related to testing documentation and process, because the automation framework enforced a standard process and comprehensive coverage.

System Performance and Load Testing: Performance testing on mainframe applications is another aspect of QA that we modernized. Historically, performance testing might have been rudimentary, due to the difficulty of simulating many concurrent users on a green-screen interface. With automation, we leveraged the capability to spawn multiple PCOMM sessions in parallel, simulating numerous users performing transactions at the same time. We also used batch job automation to simulate heavy end-of-quarter processing loads. The results were twofold:

- We identified performance bottlenecks (for example, a particular OmniScript routine that was inefficient and slowed down when processing large numbers of records). These were reported back to the development team and subsequently optimized. The performance test scripts could then confirm the improvement after fixes.
- We ensured that the modernized system could handle peak loads (such as the spike in transactions during a plan enrollment period or the processing of annual statements). The automated load tests gave confidence that response times and throughput were within acceptable ranges. If any metric was out of line, it was caught in QA rather than by end-users in production.

It is worth noting that mainframe performance testing must be done carefully to avoid impacting other environments. We scheduled these tests in isolated windows and coordinated with IT operations. The benefit of automation here is repeatability: we can execute the same load pattern on demand, which is valuable for capacity planning and for verifying that system changes (like an upgrade of Omni to a newer version) have no adverse performance impact.

Challenges and Lessons Learned: While the outcome was positive, the journey had its challenges:

- **Initial Setup Effort:** Building the automated framework required a significant upfront effort. Team members needed to learn the PCOMM API intricacies and OmniScript details. We underestimated the time to handle certain complex screens where field coordinates had to be precisely managed in the automation. The lesson is to allocate adequate time for framework development and involve team members who have both mainframe knowledge and scripting skills.
- **Maintaining Test Scripts with System Changes:** Whenever Omni was upgraded (e.g., applying a patch or moving to a new release), some screen flows or messages changed, which in turn required updates to the automation scripts (for instance, if a screen gained a new input field, the script coordinates needed adjustment). Thus, automation is not "set and forget"; it needs maintenance. We established a practice that whenever requirements changed or the system was upgraded, part of the development task was to update not just the application but also the test scripts. Involving QA early in design changes (shift-left principle) helped mitigate maintenance effort as they could adapt scripts in parallel with development.
- **False Positives/Negatives:** In early stages, our automated tests sometimes flagged errors that turned out to be script timing issues (false positives), or conversely missed verifying something important (false negatives) due to scripting

oversight. Through the pilot and refinement phase, we caught and fixed most of these. Over time, the reliability of the test suite improved. It underlined the point that test automation suites themselves need quality control and continuous improvement.

- **Human Oversight Remains Important:** Intelligent automation greatly enhanced our QA, but human insight was still crucial. QA analysts reviewed automated test results daily, investigated failures, and decided if it was a test script issue or a genuine defect. They also performed exploratory testing to cover areas not easily automated (like checking the look-and-feel of a new web interface or unusual use cases). The best outcome was achieved by combining automation for brute-force regression with human intelligence for exploratory and investigative testing. This hybrid approach ensures that creativity and intuition (strengths of human testers) complement the speed and consistency of machines.

In summary, the results of modernizing the QA framework for Omni-based systems have been very encouraging. We achieved **faster testing cycles**, broader coverage, improved defect detection, and more robust compliance verification. These improvements directly contribute to higher software quality and reduced risk in delivering retirement services to clients. The discussion above highlights that while automation brings many benefits, it requires careful implementation and ongoing management. Nevertheless, the shift from manual testing to intelligent automation in this domain appears not only feasible but essential for organizations aiming to stay competitive and compliant in today’s environment.

Diagrams and Figures

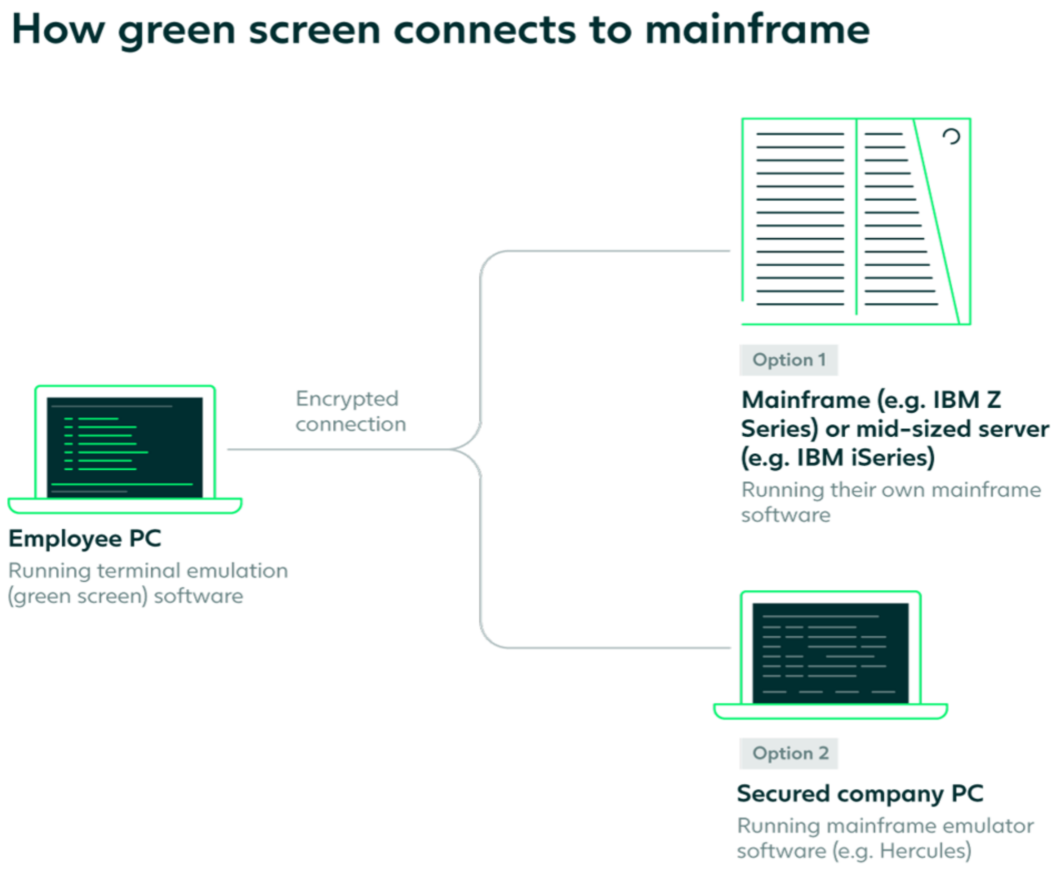
To aid understanding, we have included several **diagrams and figures** illustrating both the legacy challenges and the modernized QA solution. This section provides a brief overview of these visuals and their significance in the context of the discussion:

Figure 1 (QA Automation Framework Architecture): This diagram (presented earlier in the Methodology section) shows the blueprint of the modern QA framework. It highlights how the Test Automation Orchestrator connects to different parts of the Omni system (online interface, batch jobs, database) to execute tests in an integrated manner. The figure emphasizes the **bidirectional flow** of information: the orchestrator sends inputs (transactions or job triggers) and then retrieves outputs (screen data, database records, reports) for verification. By visualizing the components (PCOMM, JCL, DB, etc.) and their interactions, stakeholders can appreciate how the traditionally siloed testing activities are unified in one framework. This diagram was crucial for communicating the design to both QA team members and management, demonstrating that all critical facets of the system would be under automated test coverage.

Figure 2 (Manual vs Automated Testing Duration): This bar chart provides a quantitative comparison of testing efficiency, illustrating the dramatic reduction in time when moving to automation. It serves as evidence of why the investment in automation is justified. Seeing a visual comparison (40 hours vs 8 hours) resonates with project managers and executives, as it translates directly to cost and time savings. The figure also implicitly suggests how automation can enable more frequent testing (since 8 hours can feasibly be done overnight or within a workday, whereas 40 hours required nearly a week of dedicated effort previously).

Legacy Process Illustration (Terminal Connection): In the introduction, we discussed how mainframe systems are accessed via terminal emulators (green screens). *Figure 3* below provides a conceptual illustration of this: an employee PC running a terminal emulator connects over an encrypted channel to either a physical mainframe or a mainframe emulator server. We include this diagram to remind readers of the context in which Omni operates. It underscores that even though the user interface is dated (text-based), secure connectivity and centralized processing are hallmarks of mainfrarchitecture. This understanding reinforces *why* specialized automation (via PCOMM or similar) is needed, since standard web testing tools cannot directly interact with a green screen.

Figure 3: Legacy Mainframe Access via Terminal Emulator (Green Screen). Employees access Omni's mainframe-based


How green screen connects to mainframe
Encrypted connection
Employee PC
Running terminal emulation (green screen) software
Option 1
Mainframe (e.g. IBM Z Series) or mid-sized server (e.g. IBM iSeries)
Running their own mainframe software
Option 2
Secured company PC
Running mainframe emulator software (e.g. Hercules)

system using a PC terminal emulator (e.g., IBM PCOMM). The emulator provides an encrypted connection to the mainframe or mid-range server running the Omni application. This traditional setup (often called a "green screen" interface) poses challenges for QA automation, as it is not as readily scriptable as modern web interfaces. Solutions like PCOMM's API enable automated control of these sessions, which is a key enabler for modernizing QA on such legacy platforms.

By examining these figures together, one can trace the evolution: **from** a legacy world where testers manually drive green screens on individual PCs, **to** a modern orchestrated setup where a central automation engine conducts numerous virtual " testers" on green screens and batch jobs in parallel. The diagrams and figures not only depict the technical changes but also symbolize the shift in mindset – treating the mainframe not as a black box that requires human operation, but as another platform that can be tested with the same level of automation and rigor as web or mobile applications.

Case Studies

To illustrate the practical impact of modernizing QA frameworks in Omni-based retirement systems, this section presents two representative case studies. These case studies demonstrate the transition from manual testing to automated QA in real-world scenarios, highlighting the challenges faced, solutions implemented, and benefits realized.

Public Sector Pension Modernization

A large public sector pension administration system, serving over 1 million participants, was running on an Omni-based platform that had been heavily customized over decades. The organization faced mounting issues with their QA process. Each software update required enormous manual testing effort by teams of temporary contractors, leading to long turnaround times and significant backlogs. The risk was compounded by the fact that the system software was out-of-date; they were running an older version of Omni that was no longer supported by the vendor, which introduced security and compliance risks.

The primary challenge was to upgrade to the latest Omni version and ensure all existing functionality and data would continue to work correctly. Given the high liabilities (over $80 million in PII data risk and critical fund operations) at stake, failure was not an option. The QA team had to validate not only new features but also *everything* that already existed, across multiple integrated applications and OmniScripts. Doing this manually would have been nearly impossible within a reasonable timeframe.

The organization invested in developing an automated QA framework, very similar to what we described in this paper. They brought in QA consultants experienced in both mainframe testing and Omni. Key steps taken included:

- Automating core business process tests (enrollments, contributions, benefit calculations) using PCOMM scripts. These tests were able to exercise the upgraded Omni system through its new GUI as well as legacy screens.
- Utilizing Omni's API for parallel regression testing: For certain calculations, the old system and new upgraded system were both run with the same inputs via automated scripts to compare outputs (a form of *automated parallel testing* to ensure the upgrade didn't change results inadvertently).
- Extensive use of JCL batch automation to rerun end-of-year processing (generating annual statements, interest crediting, etc.) and compare the results from the new system against the prior system's outputs. Any differences were flagged for investigation – in most cases differences were due to intentional improvements or bug fixes in

the new version, but a few revealed incompatibilities that had to be addressed via code adjustments or data conversion fixes.

With the help of automation, the institution completed the Omni upgrade and migration with a high degree of confidence. The automated regression suite comprised over 500 test scenarios covering 18 months of critical operations. It was reported that core Omni calculations post-migration performed well *with no participant data issues*. The QA automation caught a number of issues during testing that, if left unnoticed, could have caused severe problems after go-live (for example, a discrepancy in loan interest calculation due to a changed rounding rule in the new version). By addressing these pre-go-live, the production rollout was smooth – critical batch jobs and quarterly statements ran without incident. This case study underlines the value of automation in de-risking a major system modernization. The investment in building a robust automated QA suite paid off by enabling a successful upgrade that aligned the organization with supported software and improved their operational stability.

After the upgrade, the organization decided to **continuously maintain** the automated test suite for ongoing QA. The initial automation became the foundation for their QA in all future enhancements. They also noted improvements in team capabilities – the QA staff upskilled in automation and mainframe scripting, reducing reliance on temporary contractors. In essence, the case showed that modernizing QA was an integral part of modernizing the platform itself.

Mid-Size 401(k) Provider – QA Efficiency and Compliance

A mid-sized 401(k) recordkeeper using OmniPlus served around 200,000 participants across various corporate retirement plans. The company was growing, but its QA processes were lagging behind. They had a small QA team that was overwhelmed by the volume of testing needed for each plan customization or new regulatory update. For instance, when new IRS contribution limits were announced annually or when the SECURE Act introduced changes in distribution rules, the QA team had to verify that the system changes correctly implemented these rules for every plan. Manual testing of these scenarios was taking too long, causing delays in rolling out updates to clients.

Key challenges included:

- **Compliance Testing Load:** Each year, the system had to be tested for updated limits (like the new 402(g) deferral limit, catch-up contribution changes, required minimum distribution age changes, etc.). Doing this for every plan variation manually was error-prone; missing a scenario could mean a plan goes out of compliance.

- **User Interface Transitions:** The provider was in the process of rolling out a new web-based front end for Omni (moving some functions from OmniStation green screens to a web portal for plan sponsors). This meant they needed to test both interfaces – the legacy and the new – for consistency. Ensuring that a transaction on the web UI resulted in the same database state as one on the green screen was critical.

- **Resource Constraints:** With only a handful of QA engineers, they could not dedicate people to run full regressions on demand. Often, only partial testing was done due to time constraints, which was a significant risk.

They decided to implement intelligent automation focusing on two angles: regression breadth and compliance depth.

- They chose a **codeless automation tool** that had mainframe support (one of the tools in the market that allows building test flows via a visual interface, integrated with 3270 terminal support). This was selected to enable their existing team (who were domain experts but not strong coders) to build automation with a relatively gentle learning curve. Codeless or low-code test automation platforms can empower QA analysts to create automated tests without writing extensive code, which was suitable for this team's profile.

- The team automated a regression suite of around 150 test cases covering common plan operations: enrollment, contributions with various sources (pre-tax, Roth, employer match), loan initiation, loan repayment, distribution processing, and an array of compliance checks. They parameterized these tests so that they could easily input different plan IDs or different limit values to re-use the scenarios across plans and years.

- For compliance, they took an interesting approach: using **Omni's compliance testing feature** (which can run tests like ADP/ACP) in an automated fashion. They wrote automation scripts to populate test data for a plan (like generating a synthetic set of employees with salaries and deferral rates that would intentionally fail a nondiscrimination test), then executed Omni's compliance module, and captured the results. By varying the input data, they could test borderline cases (e.g., just at the passing threshold vs just failing) to ensure the system's compliance logic was properly updated for the latest rules. These tests ran annually or whenever compliance-related code was changed.

The QA team saw immediate benefits. What used to take them two weeks of manual testing (for a major release with regulatory updates and some new features) was now done in 3-4 days with automation. This not only sped up the delivery to clients but also reduced stress on the small team. They reported nearly 100% coverage of required compliance scenarios after automation, whereas before they estimate they could only cover ~60-70% due to time limits. In one instance, an automated test caught a subtle defect in which the new web UI allowed an out-of-range input that the green screen would not (a validation check was missing on the web front-end). The automation script, which was set to try a range of inputs, found that and developers fixed it before release. This prevented a potential compliance issue where a plan sponsor might have entered an invalid value through the web portal.

From a compliance perspective, the provider's confidence in their system's adherence to rules increased. They began to invite their internal compliance officers to review the automated test evidence. Seeing detailed logs of how the system responds to various compliance scenarios gave those officers reassurance, which was important for corporate governance.

The success of QA automation in this mid-size firm had an interesting cultural effect: it demonstrated to management the value of investing in modernization. Encouraged by QA's improvements, they subsequently invested in automating some of their operational processes as well (e.g., using RPA for data entry tasks outside of QA). In other words, QA became a **trailblazer for innovation** within the organization, showcasing how legacy systems can be made more efficient and reliable with modern techniques.

These case studies underscore that while each organization's context differs, the fundamental gains from modernizing QA frameworks are consistent: **faster testing cycles, better coverage (especially for compliance), higher quality releases, and reduced risk**. Importantly, they also show that automation is not just about tools, but about people and process changes. In both cases, teams had to adapt – learning new skills, trusting the automation, and integrating it into their routine. The successful outcomes required management support, proper planning, and a willingness to iteratively improve the automated tests.

The lessons from these cases align well with industry reports and expert analyses, which advocate updating legacy QA processes to keep up with digital transformation efforts. As one practitioner noted, adopting test automation in mainframes and shifting testing left in the lifecycle is increasingly essential for delivering rapid, high-quality changesibm.com. Both cases above exemplify this: by bringing modern QA techniques to Omni-based systems, these organizations achieved a level of agility and reliability that would have been very hard to attain with manual testing alone.

CONCLUSION

The evolution of QA frameworks for Omni-based retirement systems from manual mainframe testing to intelligent automation marks a significant leap forward for the retirement services industry. In this paper, we presented an in-depth examination of how such modernization can be achieved and what benefits it brings. The key conclusions and takeaways are as follows:

- **Necessity of Modernization:** Legacy mainframe systems, like those running Omni for recordkeeping, are robust but were traditionally insulated from modern testing practices. We highlighted how the growing complexity of retirement products, frequent regulatory changes, and the demand for faster release cycles have made it **necessary** to overhaul QA processes. Manual testing simply cannot scale to meet these demands without compromising quality or speed.

- **Framework and Tools:** We outlined a comprehensive QA automation framework that integrates with Omni systems at multiple levels (UI, batch, API, database). Tools such as IBM PCOMM (for terminal automation), JCL scripts (for batch job control), and OmniScript (for embedding custom logic) form the pillars of this framework. The design demonstrated that even a "green screen" mainframe application can be effectively tested with modern automation techniques when the right interfaces and scripts are utilized.

- **Improvements Achieved:** The transition to intelligent automation yields **dramatic improvements in efficiency and coverage**. Empirical results showed up to 80% reduction in test execution time for full regressions and a substantial increase in test coverage, leading to earlier and more frequent detection of defects. Ultimately, this means higher confidence in each software release. Fast, repeatable test cycles also enable agile methodologies (like doing bi-weekly sprints with regression tests each time) which previously were impractical in a mainframe context.

- **Compliance and Accuracy:** One of the most important contributions of modern QA automation is in the area of compliance. Retirement systems must adhere to strict rules, and our discussion showed that automation can rigorously and consistently test compliance scenarios that humans might overlook. By doing so, it safeguards organizations against regulatory breaches and ensures that participants' accounts are handled correctly. The automated logs and evidence further strengthen audit readiness. Accuracy of complex financial calculations (interest, annuities, vesting, etc.) can be verified through dual-run or AI-assisted methods with far greater thoroughness than manual methods.

- **Performance and Reliability:** Automation not only helps functional testing but also performance and reliability testing, as seen by the ability to simulate loads and monitor system behavior under stress. This is vital for mainframe systems that typically handle huge transaction volumes (e.g., processing payroll contributions for thousands of employers simultaneously). We observed that modern testing practices can validate system performance and help fine-tune it, leading to better end-user experiences and fewer production incidents.

- **Case Study Insights:** The case studies provided concrete evidence that organizations, whether large public pensions or mid-sized 401(k) providers, can reap significant benefits from QA modernization. They also revealed that implementing such change involves overcoming initial hurdles (skill gaps, script maintenance, etc.), but with perseverance, the outcomes are transformative. Moreover, success in QA automation can have positive ripple effects across the IT department, fostering a culture of automation and innovation.

- **Intelligent Automation and Future Trends:** The term "intelligent automation" implies the use of not just scripts, but potentially AI-driven tools. While our focus was on rule-based automation, the horizon is expanding. As noted in recent research, generative AI can assist in creating test cases and scripts for legacy code. In the future, we anticipate Omni-based QA frameworks incorporating AI components—for instance, an AI tool that analyzes Omni's COBOL code changes and suggests what test cases to run, or even generates portions of OmniScript for testing automatically. This could further reduce the manual effort in maintaining the test suite and adapt to changes more quickly. Early experiments have shown promise in using AI to distill business rules and create functional test scripts automatically, which could blend nicely with the structure we implemented.

- **Human Role and Continuous Improvement:** Finally, we conclude that the role of QA professionals in this modernized landscape, though changed, is as critical as ever. Instead of performing repetitive tests, they become **stewards of quality** by designing smart tests, monitoring automated runs, analyzing failures, and directing the evolution of the test suite. Continuous improvement is key: as new features are added to the Omni system or new regulations emerge, the QA framework must be updated and refined. The commitment to keep the QA process modern and effective must be ongoing, not a one-time project.

In closing, modernizing QA frameworks for Omni-based retirement systems is a journey that aligns technical innovation with the business goal of delivering reliable, compliant, and high-quality services to plan sponsors and participants. By embracing automation and intelligent tools, organizations can turn the once onerous task of mainframe testing into a streamlined operation. This enables them to respond faster to client needs, ensure correctness in an increasingly complex rule environment, and confidently manage the mission-critical systems at the heart of retirement administration. As the industry moves forward, those who invest in such QA modernization will likely lead in trust and efficiency, setting new standards for what can be expected from retirement plan technology.

References

1. **Enterprise Iron. (2020).** ***Case Study: Omni Modernization*.**

- *Summary:* Discusses the modernization of Omni Systems to enhance operational efficiency in retirement services. Enterprise Iron

2. **Aegis Softtech. (2024).** ***GenAI Driven Automation Testing in Mainframe Modernization*.**

- *Summary:* Explores the role of Generative AI in automating testing processes during mainframe modernization. Aegis Softtech

3. **Social Security Administration. (2018).** ***IT Modernization Plan: A Business and IT Journey*.**

- *Summary:* This document outlines SSA's comprehensive strategy to modernize its IT infrastructure, aiming to enhance service delivery and operational efficiency. Office of the Inspector General+1U.S. General Services Administration+1

4. **FIS Global. (2023).** ***Retirement Solutions | Plan Administration*. *fisglobal.com***

- *Summary:* Highlights the capabilities of the Omni system in administering retirement plans efficiently.

5. **Kyndryl. (2024).** ***The Future of Mainframe Modernization with Artificial Intelligence (AI)*.**

- *Summary:* Discusses how AI can facilitate mainframe modernization by analyzing applications and automating code restructuring. testRigor+3Aegis Softtech+3Kyndryl+3

6. **Voya Retirement Insurance and Annuity Company. (2017).** ***Service Organization Control 1 Report on the Suitability of the Design and Operating Effectiveness of the Controls*.**

- *Summary:* Evaluates the effectiveness of controls within the Omni platform used for retirement plan administration. sponsor.voya.com+1FIS Global+1

7. **ACCELQ. (2024).** ***Top 8 Mainframe Testing Tools in 2025*.**

- *Summary:* Provides insights into leading tools for automating mainframe testing, including their features and benefits. accelq.com

8. **Enterprise Iron. (2022).** ***Case Study: Retirement Services Data Modeling*.**

- *Summary:* Describes the development of a canonical data model to unify data across multiple Omni legacy systems. Social Security+2Enterprise Iron+2FIS Global+2

9. Deloitte. (2023). Automation with Intelligence. (Deloitte Automation with Intelligence)

- *Summary:* Explores the integration of AI with robotic process automation to enhance business processes, including mainframe systems.

10.* IBM. (2024). *IBM Unveils watsonx Assistant for Mainframe Mastery*. *(futurumgroup.com)

- *Summary:* Introduces IBM's AI assistant designed to bridge the mainframe skills gap and enhance operational efficiency.

11.* Kyndryl. (2025). *Mainframes — Services and Solutions*. *(kyndryl.com)

- *Summary:* Discusses AI-based automation tools for mainframe modernization, including code analysis and restructuring.

12.* testRigor. (2025). *Mainframe Testing - testRigor AI-Based Automated Testing Tool*. *(testrigor.com)

- *Summary:* Highlights AI-powered, no-code test automation spanning mainframe and other systems.

13.* IBM. (2024). *IBM Application Performance Analyzer for zOS*. *(IBM Application Performance Analyzer)

- *Summary:* Provides tools for analyzing and optimizing application performance on mainframe systems.

14.* Macro 4. (2024). *Why Galasa is a 'Big Deal' for Mainframe Testing.(planetmainframe.com)

- *Summary:* Explains the significance of the Galasa framework in automating mainframe testing processes.

15.* T-Systems. (2025). *Mainframe Modernization and Automation Services.(t-systems.com)

- *Summary:* Offers insights into modernizing mainframe systems with automation and AI technologies.

16.* Accenture. (2023). *AI and Automation in QA: What the Future Holds.(accenture.com)

- *Summary:* Discusses the impact of AI and automation on quality assurance processes across industries.

17.* Deloitte. (2022). *The Future of Mainframe Modernization in Financial Services*. *(deloitte.com)

- *Summary:* Examines the role of mainframe modernization in the financial sector, emphasizing automation and AI.

18. **Kyndryl. (2024).** ***AI-Aided Mainframe Modernization*****.** ***kyndryl.com***

- *Summary:* Explores how AI assists in various phases of mainframe transformation, including code analysis and testing.

19. **IBM. (2024).** ***IBM watsonx Orchestrate for Mainframe Management*****.** ***(IBM MediaCenter)***

1. *Summary:* Introduces AI-driven orchestration tools for managing mainframe operations efficiently.