International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 5, Issue 1, 92-100, 2024
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V5I1P111

Original Article

The Importance of Testing and Validation in Legacy Application Upgrades

Arun K Gangula
Independent Researcher.

Abstract *- Organizations face growing difficulties with their legacy systems because these systems have become outdated and require more money to maintain while exposing security risks. The process of modernizing these systems requires essential changes but creates substantial risks that affect operational stability and data protection and introduce security weaknesses. A multi-layered testing and validation framework serves as the main risk reduction method for protecting legacy system upgrades according to this research. The proposed method implements testing as an ongoing process which runs across the entire modernization lifecycle instead of using it as a traditional quality assurance check at the end. The framework includes three testing categories which consist of functional validation for data integrity and regression and integration testing and non-functional validation for performance and security testing and user acceptance testing for user-centric validation. The combination of continuous thorough testing according to technical studies and business examples helps organizations handle complex situations while minimizing risks to achieve successful system transitions. The research demonstrates that organizations need to implement proactive validation methods to convert dangerous modernization projects into controlled initiatives that deliver value.*

Key words *- Legacy Systems, Software Modernization, Software Testing, Validation, Regression Testing, Integration Testing, Data Migration, Risk Management, User Acceptance Testing (UAT).*

1. Introduction

1.1. The Paradox of Legacy Systems: Indispensable yet Obsolete

Organizations maintain essential outdated technology systems and applications and architectural frameworks which they need to operate their business. The systems continue to exist because they contain vital business knowledge and data accumulated throughout multiple decades [1]. The systems operated as advanced technology in their time but now use outdated monolithic structures and programming languages including COBOL and FORTRAN [2] [3]. The systems fail to support contemporary paradigms including cloud computing and APIs and microservices which make them necessary for business expansion yet limit organizational development. A system becomes legacy based on its inability to fulfill contemporary requirements for mobility and interoperability and responsiveness rather than its age. The concept of instant legacy transforms system evolution into an ongoing process which focuses on developing long-term adaptability through continuous modernization efforts.

1.2. The Modernization Mandate: Drivers and Inherent Perils

The combination of high maintenance expenses and weak system performance and scattered data and increasing security threats leads to modernization needs [4]. The WannaCry ransomware attack and other incidents demonstrate how unsupported platforms with unpatched vulnerabilities create severe security threats for organizations. Organizations face dual risks from non-compliance with GDPR and HIPAA regulations because they must deal with both legal consequences and financial penalties [4]. Drivers for modernization exist but the process brings major risks to the table. The failure of modernization initiatives results in system breakdowns and data destruction and excessive project expenses [1]. Business leaders tend to focus on the apparent dangers of acting, but they fail to recognize the escalating threats that come from staying idle which leads to indecisiveness [2].

2. Taxonomy of Risks in Legacy Upgrades

The process of updating legacy applications through modernization creates multiple connected security risks which affect operational and technical aspects as well as business and financial operations and security and compliance requirements. The failure of any domain within these three risk categories leads to a chain reaction that could end in project failure and major organizational disruptions.

2.1. Operational and Technical Risks: Disruption and Data Integrity

The first risks that emerge from legacy system upgrades stem from operational and technical aspects. Legacy systems maintain their brittle nature because of unrecorded modifications and tightly connected components and complex code organization that has

developed over time. Small system modifications create unpredictable effects which spread across the entire system. The main operational challenge during migration or cutover periods involves extended system downtime which interrupts vital business operations. The integration of new components with legacy elements becomes challenging during modernization because it creates hybrid architectures that require technical integration between different system parts. The system experiences higher risks of data transmission problems and stability issues because integration points depend on unreliable file-based transfers instead of stable APIs. The new system fails to match or surpass legacy system performance levels even though it runs on contemporary infrastructure platforms.

Data integrity stands as a vital risk factor that organizations need to address. The process of extracting legacy data for transformation and loading into modern platforms creates high risks of data loss or corruption because historical data exists in multiple formats and duplicate entries. The discovery of post-migration issues with reporting accuracy and operational performance becomes possible after system migration [2].

2.2. Security and Compliance Risks in Modern Environments

The security risks of legacy systems are substantial because they were built before modern cyber threats became a concern. The systems run on obsolete platforms with unsecured third-party libraries because security updates for these components have become unavailable [4]. The practice of insecure coding continues to persist using hardcoded credentials and weak encryption methods [2]. The security constraints of these systems create difficulties for organizations to comply with regulations. The General Data Protection Regulation (GDPR) and Health Insurance Portability and Accountability Act (HIPAA) establish strict data handling and auditing standards which most legacy systems fail to fulfill [4] [5].

The healthcare sector faces a severe problem because outdated medical system software endangers both system security and patient wellness [6]. The transition to modern systems brings forth fresh security risks that need to be addressed. The process of transferring sensitive information to cloud environments becomes dangerous when security protocols are not properly implemented [7]. The integration of modern third-party services creates additional security risks because unprotected APIs and external interfaces become vulnerable entry points.

2.3. Business and Financial Risks of Modernization Failure

Legacy modernization project failures create major financial and business challenges for organizations. The direct financial loss from capital investment failure amounts to millions of dollars while operational downtime generates additional revenue losses. The 2004 Comair legacy crew-scheduling system failure resulted in flight cancellations that caused major financial damage to the company. The implementation of modernization projects leads to negative impacts on organizational reputation. System security breaches and instability that expose customer information will damage both customer trust and their loyalty levels [5].

Organizations fail to accurately calculate modernization expenses because they do not properly handle their existing technical debt. The accumulation of technical debt from multiple decades of short-term solutions requires complete system refactoring and redesigning to become debt-free. Projects which fail to address this requirement will experience budget increases and time delays according to [5] and [7].

The main reason behind these risks stems from insufficient knowledge about the system. The combination of employee departures and insufficient documentation leads to progressive loss of institutional knowledge. The majority of essential system knowledge exists only in the minds of a few experienced workers who possess what experts call "tribal knowledge" [8]. The absence of system understanding makes it difficult to create reliable plans and perform accurate risk evaluations and cost predictions. The implementation of cloud adoption and microservices brings new security challenges to the modernization process.

The transition to distributed systems becomes unsuccessful when organizations lack proper skills to handle cloud security and data sovereignty and operational complexity [7]. The testing strategy needs to fulfill two essential purposes which include verifying that legacy business operations remain intact and confirming proper system behavior within the new operational framework. The lack of preparedness among traditional testing teams to handle both legacy system preservation and new operational requirements leads to higher chances of project failure [9].

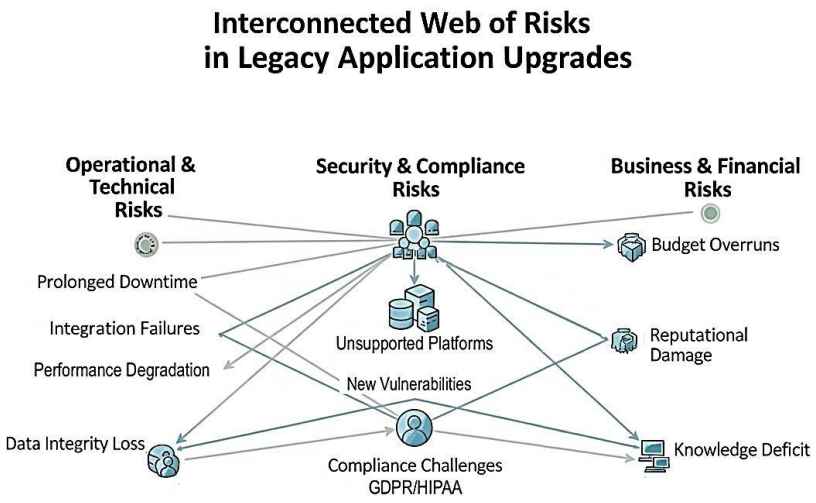

Fig 1: Interconnected Web of Risks in Legacy Application Upgrades

3. Multi-Layered Framework for Functional Validation

A comprehensive framework for functional validation exists to handle legacy upgrade risks through multiple stages. The framework starts with data integrity assessment which must be completed before moving on to system integration testing for each successive stage. This method allows organizations to reduce risks through controlled and sequential modernization steps.

3.1. The Foundational Layer: Data Integrity and Migration Testing

The foundation of any legacy system depends on data integrity which requires initial validation efforts. The process of data migration requires multiple steps which start with extraction followed by transformation and cleansing before validation. The prolonged operational use of legacy systems has created fragmented data that contains duplicated information and inconsistent values [2].

Structured, three-phase validation process is required [10]:

- Pre-Migration Validation: The first step of this phase requires analyzing source data to detect both data quality problems and duplicate information and inconsistent data points. The process of data cleansing should start at the source location because it will make all following operations more straightforward.
- Migration Validation: The migration process requires validation to verify that transformation rules are executed properly. The process uses representative data sets to validate both schema mapping accuracy and transformation precision. The target system output needs to exactly match the original source data records for maintaining complete data fidelity [2].
- Post-Migration Validation: The system performs complete validation procedures following migration operations. The system needs automated integrity checks and reconciliation reports to confirm that all records match and financial totals remain accurate, and data remains c o r r e c t . The large data volume requires automated data-element level checks to identify small errors which include rounding discrepancies and data truncation problems [11].

3.2. The Stability Layer: Advanced Regression and Characterization Testing

The confirmation of data integrity leads to the validation of functional stability. The process of regression testing serves as a critical requirement to verify that essential business operations continue working properly after system modifications. This involves:

- Test Suite Creation and Selection: The development of a regression test suite requires either new case creation or selection of essential system functionalities from existing cases.
- Prioritization: The selection of test cases depends on risk levels and criticality and how often users need to perform them. The complete execution of the entire test suite becomes impractical for each system modification.
- Execution and Maintenance: The test execution process reveals system regressions which developers need to resolve. The test suite needs regular maintenance to incorporate new features and eliminate outdated test cases.

Systems without documentation require Characterization Testing which also operates under the name Golden Master Testing.

The technique documents present system operations to create a reference output for future reference. The system runs the same inputs after migration to check for any unexpected output modifications.

The testing method now concentrates on proving system behavior matches the original system instead of checking for accuracy. The testing process consists of two distinct stages.

- Phase 1: The modernized system needs regression and characterization testing to verify it duplicates legacy functionality including all documented system bugs.
- Phase 2: The team can proceed with controlled legacy issue fixes after confirming system equivalence.

The method protects current system operations from damage while allowing a systematic approach to fix documented system problems.

4. The Cohesion Layer: Integration Testing For Hybrid Architectures

The process of legacy modernization creates systems that combine different architectural elements. The testing process of integration verifies that new system components work properly with existing legacy system modules. The process of integration testing requires developers to unite different software modules before checking their operational connections. The integration of legacy systems becomes complicated because they do not have APIs, and their interfaces are unstable, and their data formats differ from one another [8].

The selection of integration approaches depends on both system design structure and project danger level. The original work provides a comparison of these strategies which appear in Table I. Large-scale regression management heavily depends on automation systems for its operation [12]. Legacy systems face difficulties when integrating with contemporary test automation tools because their interfaces remain unstable and their architectural design imposes limitations. A testing approach that combines different methods must be implemented because of these system limitations.

4.1. API-level automation for new services.

Specialized UI automation tools which include image-based systems and object model hooking methods should be used for testing legacy front-end interfaces. The testing process requires manual inspection of complex workflows which cannot be automated at a reasonable cost. The successful modernization of systems depends on detailed planning for testing diverse environments which need various tools and expert skills.

Table 1: Comparative Analysis of Integration Testing Strategies in a Legacy Modernization Context

Strategy	Description	Best Suited For (Legacy System Profile)	Advantages	Disadvantages/Risks
Big Bang	All components are integrated at once and tested as a single unit.	Small, simple systems with few interdependencies. Not generally recommended for complex legacy systems.	It can be quick for very small systems.	It is extremely difficult to isolate the source of faults. High risk of complete system failure. Late discovery of interface defects.
Top-Down	High-level control modules are tested first. Lower-level modules are simulated using "stubs."	Systems with complex business logic and control flow at the top. Projects where validating the overall architecture early is a priority.	Allows for early validation of major design decisions and system workflows.	Significant functionality in lower-level modules is tested late in the cycle. Writing and maintaining complex stubs can be time-consuming.
Bottom-Up	Low-level, foundational modules (e.g., data access, calculations) are tested first. Higher-level modules are integrated progressively using "drivers" to simulate calls.	Systems where the reliability of foundational data processing, utility functions, or external interfaces is the highest risk.	Critical, low-level bugs are found early. Facilitates testing of individual components in isolation.	High-level system behavior and user interface flaws are discovered late. The system does not exist until the very end.

5. Validating Critical Non-Functional Requirements

The modernized system needs to pass functional validation tests to execute necessary operations, but non-functional requirement (NFR) validation checks its performance levels and security and reliability standards. Legacy modernization projects

frequently focus on enhancing non-functional requirements (NFRs) as their main reason for advancement. The validation of these requirements stands as a core indicator which determines project success.

5.1. Performance Validation: Ensuring Scalability and Responsiveness

The main reason organizations choose modernization is to solve performance and scalability problems that exist in their outdated systems. The validation process for performance becomes essential because it verifies the modernized system achieves its performance targets. The performance validation process includes various specific performance testing methods. Load Testing: The test evaluates system performance under normal usage conditions and maximum user activity levels. The system performance evaluation includes response time measurements and throughput analysis and resource consumption monitoring to verify operational stability at expected usage levels.

Stress Testing: The testing approach exceeds system operational boundaries to detect critical failure areas and evaluate system reactions during maximum stress conditions. The testing method shows system design constraints and maximum resource capacity and system failure management capabilities. Endurance (Soak) Testing: The process of running tests for extended periods under normal usage conditions helps detect memory leak problems that only become visible during longer testing sessions. The transition of systems to continuous cloud operation requires this testing method to be implemented.

A performance validation strategy requires first establishing a baseline from the current system which will function as a reference point for comparison. The next step requires defining performance targets which should include metrics such as "transaction response times should stay under 2 seconds when 1,000 users access the system." The testing process requires developers to create authentic user simulation scenarios while tracking CPU performance and memory usage and disk operations and network data transfer activities throughout the test execution [13].

5.2. Security Validation: From Vulnerability Assessment to Penetration Testing

The process of modernization enables organizations to build security into their system architecture during initial development which solves the security problems found in outdated systems. The transition process creates fresh security threats which organizations need to handle. Security validation needs to run continuously across all stages of modernization to follow the Security-by-Design approach for risk reduction.

5.3. Complete security validation system requires these essential elements

Threat Modeling and Vulnerability Assessment: The first step of planning requires threat modeling to detect security risks that will appear in the new system architecture. The assessment of legacy system vulnerabilities produces a complete list of security problems which need resolution.

- Static and Dynamic Code Analysis: The development process requires organizations to use Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) tools. The source code analysis of SAST identifies known vulnerabilities but DAST monitors application behavior during runtime [10].
- Penetration Testing: The deployment readiness check requires complete penetration testing to be performed. The testing method duplicates actual cyber-attacks to detect security weaknesses that exist in applications and their configurations and infrastructure base.
- Compliance Validation: Organizations that need to meet regulatory standards must perform a formal compliance audit as their final step. The audit process verifies that organizations follow HIPAA and PCI-DSS and GDPR standards by confirming all required security controls are properly implemented [4].
- Modernization projects benefit from non-functional testing which fulfills two essential functions. The new system quality verification process through non-functional testing also helps identify weaknesses in the existing system infrastructure. The results of stress testing the legacy system before migration will show performance weaknesses which support the need for modernization. The process of early vulnerability assessments produces security problem lists which guide project planning and return on investment evaluation.
- Cloud-native architecture brings substantial changes to the way organizations perform non-functional testing. The testing process now requires evaluation of distributed systems which operate on short-lived cloud-based infrastructure instead of fixed hardware platforms. The new performance testing requirements include verification of automatic scaling rules and measurement of data transfer speed between different regions. Security testing needs to verify both cloud infrastructure settings including firewalls and identity controls and the complete integrity of the CI/CD pipeline [14] and [7].

The changing landscape requires testing teams to acquire new skills at a substantial level. Testing teams need to master cloud platforms together with containerization tools and infrastructure-as-code systems to perform non-functional testing effectively in modernized environments.

6. The Critical Role of User Acceptance and Adoption

A system migration with perfect technical execution of data accuracy and functional stability and performance will fail if users refuse to use the new system. The last part of the validation framework concentrates on human aspects to verify that the modernized system operates effectively for business needs and users will actually use it throughout the organization.

6.1. User Acceptance Testing (UAT as the Ultimate Arbiter of Success

User acceptance Testing (UAT) serves as the essential validation stage that precedes system deployment because it represents the last phase of system testing. End-users conduct system evaluation through actual business operations to verify the system meets its original modernization requirements [9]. The evaluation at this stage checks both system operational capabilities and its ability to maintain business operations in real-world scenarios.

The process of User Acceptance Testing requires a defined structure to guarantee both reliability and complete assessment results. The process contains these essential elements:

- Defining Acceptance Criteria: Business stakeholders need to define specific measurable criteria before starting any testing process.
- Developing Test Scenarios: The business analysts work together with power users to develop complete test scenarios which represent both regular business operations and scheduled tasks.
- Conducting Test Sessions: The test cases run through a controlled environment that mimics production operations by users who belong to all relevant departments.
- Feedback and Defect Management: The project team tracks down defects and usability problems which they then prioritize for resolution.
- Formal Sign-Off: The business stakeholders give official deployment authorization after the system meets all acceptance criteria, and they resolve essential problems.

The process of UAT becomes more important during legacy system modernization initiatives. The combination of insufficient documentation and disappearing institutional knowledge makes long-serving users who participate in UAT the only remaining source of operational and business-rule understanding. Users who notice discrepancies in past functionality tend to reveal important undocumented procedures instead of showing resistance to change. The UAT process needs to function as both a validation step and a requirements discovery phase which occurs near the end of development. The process requires longer durations and skilled business analysts and adaptable development methods that can accept vital new functionality discoveries.

6.2. Overcoming User Resistance and Ensuring Business Alignment

User resistance can defeat even the most successful technical system migrations. Users tend to resist new systems because they already know their current workflows even though these systems offer better efficiency [15]. Research shows that system implementation failures mostly result from poor user adoption strategies instead of technical system problems [16]. The proper execution of UAT functions as a fundamental tool for managing organizational changes. User involvement at all stages of the process leads to higher levels of user engagement and ownership [7]. Users who participate in the development process through input will become more supportive of changes because they evolve from system recipients to system contributors.

The feedback obtained during UAT provides information that goes beyond defect identification. The process helps identify complex workflow areas and inefficient processes which enable developers to create better training materials and documentation. The new system implementation will achieve better operational alignment and user acceptance after deployment because of this process. The success of UAT depends on having accurate data as its foundation. Users will focus on identifying data quality problems instead of evaluating system functionality when they need to test with faulty or missing or improperly formatted data. The process of UAT becomes less effective while users lose confidence in the system when they encounter these issues. A mandatory Data Integrity Sign-off procedure needs to exist before starting UAT operations. The technical teams must offer stakeholders documented evidence that the UAT environment contains precise and whole and dependable data.

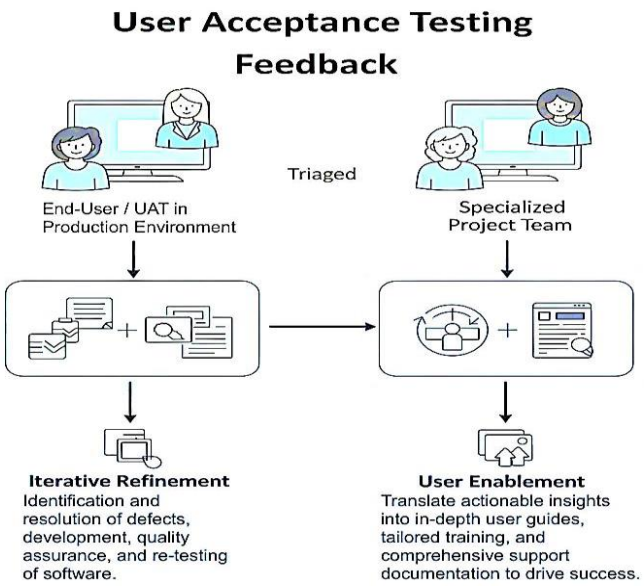

Fig 2: The UAT Feedback Loop for Iterative Refinement and User Enablement

7. Analysis of Real-World Implementations: Case Studies and Lessons Learned

Real-world modernization programs deliver functional knowledge that goes beyond academic theories. Multiple industrial case studies demonstrate recurring patterns of success and failure which provide actionable knowledge for upcoming modernization projects.

7.1. Insights from the Banking and Finance Sector

The banking and finance industry functions as an essential environment to test legacy system modernization initiatives. The financial institutions maintain mission-critical legacy systems based on mainframes to execute secure and reliable transaction processing at high volumes [17]. Major retail banks perform modernization projects based on three main factors which include cost reduction for IT operations and COBOL programmer shortage and the need for omni-channel customer solutions [17].

The sector implements migration strategies through three main components which include core application redesign and non-critical system maintenance and COTS product implementation for commodity functions [17]. These implementations demonstrate that project success depends heavily on proper governance and effective prioritization methods. The delivery of business value faces delays when centralized models use core architecture teams for prioritization because they fail to understand specific domain requirements. Domain-driven approaches that operate independently from central control achieve better business requirement understanding but create architectural problems and duplicate work. The most successful strategies unite domain- level execution with centralized architectural direction to achieve both operational unity and performance excellence [17].

7.2. Challenges and Successes in Healthcare and Public Sector Modernization

Healthcare and public sector modernization projects face special limitations because they need to follow strict regulatory and compliance standards. The sectors require absolute data integrity and legal record admissibility because they maintain critical information [11]. The Calgary Police Service conducted a major project to transfer about four million criminal case files into a new system. The project followed ISO 13008 standards for digital records conversion and migration to guarantee the authenticity and legal protection and reliability of all transferred data. The project achieved success through extensive testing and complete documentation and effective teamwork between RIM and IT personnel.

The research included IT architects from major healthcare organizations and financial institutions who provided qualitative data about their experiences. The research identified three essential factors which led to successful outcomes through effective stakeholder partnerships and complete documentation systems and ongoing technical education for staff members. The research demonstrates that organizational discipline together with effective communication stands equally important to project success in complex regulated environments as does technical execution.

7.3. Synthesized Lessons for Future Modernization Endeavors

Multiple post-implementation assessments of cross-industry projects have identified fundamental lessons which guide successful legacy modernization initiatives [18].

- Meticulous Planning and Assessment: The main reason behind project failures stems from insufficient planning at the beginning of the project. Technical approaches to projects that lack understanding of business requirements and hidden system dependencies and undocumented system logic tend to fail. The discovery phase needs to be extensive because it serves as a fundamental requirement for project success [18].
- Address Data Quality Early: All legacy systems contain data that is considered "dirty" in nature. The project team needs to recognize dirty data at the start and dedicate funds to cleaning the data in its original systems before migration occurs. The process requires automated data validation and reconciliation to function as essential components.
- Recognize the Human Factor: The main obstacles during migration projects stem from organizational aspects instead of technical aspects. The main barriers to success include employee resistance to change and insufficient stakeholder participation and insufficient preservation of organizational knowledge. The success of projects depends on two essential factors which include documenting experienced staff knowledge and obtaining stakeholder support from the beginning [1].
- Strategic Tool and Partner Selection: The combination of modern tool chains with automated testing and static code analysis capabilities reduces migration risks while speeding up the process. Organizations need to exercise caution when using third-party software by selecting open-source and modular solutions because they want to prevent creating new technical problems [19].
- Adopt an Iterative Approach: The deployment of system changes through phased or iterative approaches proves more effective and secure than implementing all changes at once through a "big bang" cutover. The deployment of smaller increments allows teams to contain damage from problems while they learn from each deployment step.

The main reason behind project failures stems from uncontrolled changes to project scope. Organizations try to use modernization projects to resolve all past problems and add broad new capabilities during these initiatives. Project leaders who want success need to establish a specific and protected scope that includes the main goal of decommissioning the mainframe system while postponing additional goals like complete process transformation. The testing approach needs to maintain equal focus on validating predetermined results to achieve scheduled value delivery at budgeted costs.

8. Conclusion

Software engineering faces its most challenging and dangerous task in the process of modernizing outdated legacy applications. These essential business systems present a contradictory situation because they maintain their critical operational value yet show signs of technological obsolescence. The need for modernization has become essential because organizations face rising operational expenses and security threats and must adapt to digital transformation requirements. The research demonstrates that testing and validation should function as the fundamental base for modernization initiatives because they protect against diverse technical and security and business-related risks. The successful approach to testing involves moving beyond its traditional role as a final quality assurance check to become an ongoing proactive risk reduction process.

The paper introduces a multi-level validation system to support this new testing approach. The framework starts with data integrity verification through complete migration testing at the data level. The framework advances to functional validation through sophisticated regression and characterization methods which protect against unexpected side effects in sensitive legacy code systems. The process of structured integration testing verifies that hybrid architectures developed during phased transitions maintain their consistency. The system's operational standards at an enterprise level are validated through non-functional testing which checks performance alongside scalability and security. User acceptance testing (UAT) connects technical project outcomes to actual business requirements by promoting system adoption and confirming business objectives.

Real-world implementation experiences confirm the research results. The success of projects depends on detailed planning and immediate data quality focus and strong cross-team collaboration and step-by-step execution approach. The projects maintain strict control over their scope boundaries while avoiding excessive work commitments and maintain constant awareness of organizational changes needed for technical transformation. The process of modernization requires more than just replacing outdated technology systems. The strategic initiative transforms operational liabilities into long-term assets by developing systems which fulfill present operational requirements and maintain future readiness through security and adaptability. The success of modernization initiatives depends on testing and validation as their fundamental success factors. Systematic testing practices enable organizations to develop the necessary confidence and control and business understanding which turns dangerous modernization projects into enduring competitive advantages.

References

[1] H. A. Bakar, R. Razali, and D. I. Jambari, "A Qualitative study of legacy Systems Modernisation for Citizen-Centric Digital Government," *Sustainability*, vol. 14, no. 17, p. 10951, Sep. 2022, doi: 10.3390/su141710951.

[2] Talend, "What is a Legacy System?," *Talend - a Leader in Data Integration & Data Integrity*. https://www.talend.com/resources/what-is-legacy-system/

[3] N. Kulkarni, "LEGACY SYSTEMS AND THEIR USE IN STRATEGIC PLANNING," [Online]. Available: https://www.researchgate.net/publication/354683403_LEGACY_SYSTEMS_AND_THEIR_USE_IN_STRATEGIC_PLANNING

[4] IR Team, "What are Legacy Systems and Why Do Companies Still Use Them? | IR." https://www.ir.com/guides/what-are-legacy-systems-and-why-do-companies-still-use-them

[5] Mudunuri L.N.R.; (December, 2023); "AI-Driven Inventory Management: Never Run Out, Never Overstock"; International Journal of Advances in Engineering Research; Vol 26, Issue 6; 24-36

[6] T. Tervoort, M. T. De Oliveira, W. Pieters, P. Van Gelder, S. D. Olabarriaga, and H. Marquering, "Solutions for mitigating cybersecurity risks caused by legacy software in medical Devices: A scoping review," *IEEE Access*, vol. 8, pp. 84352–84361, Jan. 2020, doi: 10.1109/access.2020.2984376.

[7] P. K. Maroju, "Empowering Data-Driven Decision Making: The Role of Self-Service Analytics and Data Analysts in Modern Organization Strategies," International Journal of Innovations in Applied Science and Engineering (IJIASE), vol. 7, Aug. 2021.

[8] M. Srinivas, G. Ramakrishna, K. R. Rao, and E. S. Babu, "Analysis of Legacy System in Software Application Development: A Comparative survey," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 6, no. 1, p. 292, Feb. 2016, doi: 10.11591/ijece.v6i1.8367.

[9] R. Khabouze, "Modernization of legacy information technology systems," 2022. [Online]. Available: https://scholarworks.waldenu.edu/dissertations

[10] V. Duvvur, "Ensuring security compliance in legacy System Modernization: a balancing act for enhanced protection," *Journal of Mathematical & Computer Applications*, pp. 1–4, Jan. 2023, doi: 10.47363/jmca/2023(2)154.

[11] U. Fox, "Migrating Legacy Records – A case Study – ARMA Magazine," Mar. 12, 2019. https://magazine.arma.org/2019/03/migrating-legacy-records-a-case-study/

[12] S. Pargaonkar, "A comprehensive review of performance testing methodologies and best practices: Software Quality Engineering," *International Journal of Science and Research (IJSR)*, vol. 12, no. 8, pp. 2008–2014, Aug. 2023, doi: 10.21275/sr23822111402.

[13] "Legacy System Modernization: Strategies for a Digital Future," *rinf.tech*. https://www.rinf.tech/legacy-system-modernization-strategies-for-a-digital-future/

[14] P. Zheldak, "How to integrate legacy systems and modern software," *Acropolium*, Dec. 01, 2023. https://acropolium.com/blog/legacy-systems-integration/

[15] H. E. Hayretci and F. B. Aydemir, "A multi case study on legacy system migration in the banking industry," in *Lecture notes in computer science*, 2021, pp. 536–550. doi: 10.1007/978-3-030-79382-1_32.

[16] S. McDowell, "Overcoming the risks & complexities of legacy data migration," 2021. [Online]. Available: https://www.purestorage.com/content/dam/pdf/en/white-papers/wp-moor-overcoming-risks-complexities-legacy-data-migration.pdf

[17] "5x Lessons Learned from Migrating a Large Legacy to .NET 5/6 - NDepend Blog," *NDepend Blog*, Oct. 26, 2021. https://blog.ndepend.com/5x-lessons-learned-from-migrating-a-large-legacy-to-net-5-6/