Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Optimization on Test Strategy on us Healthcare Claims

Praveen Kumar Rawat
Master's in Computer Applications, PAHM, PSM, ISTQB, MCDBA
Email: Praveen.rawat1@gmail.com
Independent Researcher, Virginia, US

ARTICLE INFO

Received: 15 Jan 2023

Accepted: 05 Feb 2023

ABSTRACT

The U.S. healthcare claims processing ecosystem is a highly intricate environment involving eligibility checks, benefit verification, provider adjudication, regulatory compliance, and real-time data exchange. Traditional testing strategies often fall short in delivering the scalability, accuracy, and speed required to validate the multitude of claim scenarios in such systems. This necessitates the development of an optimized, intelligent testing strategy tailored to the dynamic and regulatory-driven nature of healthcare IT. This abstract outline an optimized test strategy focused on improving the efficiency, coverage, and reliability of testing within U.S. healthcare claims platforms. The proposed approach integrates risk-based testing with modular test design, orthogonal array combinatorial techniques, and automation-first principles. Key emphasis is placed on identifying critical claim workflows such as inpatient/outpatient processing, pre-authorization logic, secondary billing, and EDI (837/835) transaction validation. These areas are prioritized based on business impact, error likelihood, and regulatory exposure. The optimization framework incorporates test data management solutions to simulate real-world payer-provider interactions using anonymized, HIPAA-compliant datasets. Automated test suites are developed using tools such as Selenium, Postman, and Robot Framework, integrated with CI/CD pipelines (Jenkins, GitLab CI) to ensure continuous regression and smoke testing with each release. Advanced analytics and dashboards (e.g., TestRail, Grafana) monitor test performance metrics including defect leakage rate, test case reusability, and coverage ratio. Feedback loops are embedded for iterative refinement, enabling the strategy to adapt to regulatory updates like CMS rule changes or ACA reforms. In conclusion, the optimized test strategy delivers measurable improvements in test cycle efficiency, early defect detection, and compliance assurance. It empowers healthcare organizations to deploy claim processing systems that are not only functional and scalable but also resilient to operational and regulatory risk—ultimately contributing to better patient outcomes and payer-provider collaboration.

Keywords: Test Optimization, Healthcare Claims, Automation, Compliance, Quality Assurance

INTRODUCTION

The U.S. healthcare claims processing environment is characterized by high complexity, strict regulatory oversight, and ever-evolving payer-provider dynamics. Claims systems must accurately handle a wide array of processes such as eligibility verification, benefit adjudication, provider credentialing, pre-authorizations, coordination of benefits, and electronic data interchange (EDI). These systems must also comply with mandates from the Centers for Medicare and Medicaid Services (CMS), the Health Insurance Portability and Accountability Act (HIPAA), and the Affordable Care Act (ACA). Given this intricacy, ensuring software quality through a robust testing strategy is not merely desirable—it is essential.

Traditional testing methodologies often struggle to keep pace with the rapid change cycles in healthcare claims platforms. As payers continuously roll out updates to meet regulatory changes or market demands, Quality Assurance (QA) [1-3] teams are burdened with validating hundreds of configurations, logic rules, and workflows. Manual testing and static regression suites frequently result in increased defect leakage, delayed releases, and compliance risks. Therefore, an optimized test strategy is required—one that is agile, data-driven, and scalable. An optimized testing strategy for healthcare claims systems involves risk-based prioritization, modular test case design, and the strategic use of automation and combinatorial techniques such as Orthogonal Array Testing (OATS). This approach identifies high-

Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

impact claim flows—such as high-dollar inpatient services, emergency room claims, and secondary billing—as primary targets for focused testing. Risk scoring is used to classify modules based on their error probability and business importance, ensuring that the most critical areas are tested with the highest rigor [4].

Moreover, the use of automated regression suites integrated into CI/CD pipelines (e.g., Jenkins, GitLab, Azure DevOps) significantly reduces execution time and increases test frequency. Tools like Selenium for UI automation, Postman for API testing, and Robot Framework for end-to-end validations enable continuous validation across environments. Real-time dashboards powered by platforms like Allure, TestRail, or Grafana provide visibility into coverage, pass/fail trends, and defect density. Another pillar of optimization is test data management. U.S. healthcare systems must handle sensitive patient information while simulating realistic payer-provider interactions. Leveraging HIPAA-compliant, anonymized test datasets ensures that testing environments reflect production behaviour's without compromising security or privacy [5].

Incorporating feedback loops and analytics-driven improvements also ensures that the test strategy evolves continuously. Defects identified in production are analysed for root causes, and test cases are adjusted accordingly. As new CMS billing codes, plan types, or adjudication rules are introduced, the test matrix is recalibrated to reflect these changes—ensuring long-term strategy sustainability. In summary, optimizing the test strategy for U.S. healthcare claims processing systems is a strategic necessity. It enables payer organizations to achieve faster releases, better regulatory compliance, higher system reliability, and improved patient and provider satisfaction. Through a combination of automation, analytics, risk-based targeting, and continuous improvement, this optimized strategy ensures that claims platforms remain resilient, scalable, and future-ready [6].

RELATED WORK

Optimizing testing strategies in U.S. healthcare claims systems has garnered increasing academic and industry attention due to the growing need for reliable, scalable, and regulatory-compliant software solutions. As healthcare delivery and payment models become more complex, testing methodologies must evolve to ensure data accuracy, process integrity, and security compliance. Several studies, frameworks, and industrial best practices have contributed to shaping the current landscape of claims system testing optimization. This section presents a synthesis of key related work across four core areas: traditional QA limitations, automation frameworks, combinatorial testing, and healthcare-specific test data management.

1.1 Limitations of Traditional QA in Healthcare IT

Earlier research by [7] highlighted that traditional test strategies used in payer systems—primarily manual and script-based regression—were insufficient for dynamic, rule-heavy systems. These methods suffered from low test coverage, high execution time, and an inability to detect interdependent logic failures. Studies revealed that static test cases, once created, often became outdated with rapid system changes introduced by ACA mandates or CMS interoperability updates. This observation led to industry-wide interest in adopting agile and DevOps-friendly approaches for QA in healthcare environments.

2.2 Test Automation and CI/CD Integration

Substantial improvements in testing speed and reliability were observed with the integration of automated testing frameworks into DevOps pipelines. [8] proposed a CI/CD-based test framework using Selenium and RestAssured to validate API endpoints and UI flows in real-time claims processing applications. Their model demonstrated a 55% reduction in test cycle time and improved early defect detection. In payer organizations, automation has been widely adopted for core processes such as 837/835 EDI file validations, eligibility transaction testing (270/271), and real-time authorization workflows. Integration with tools like Jenkins, GitLab CI, and Azure DevOps has proven to improve traceability and accelerate feedback loops for agile teams.

Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

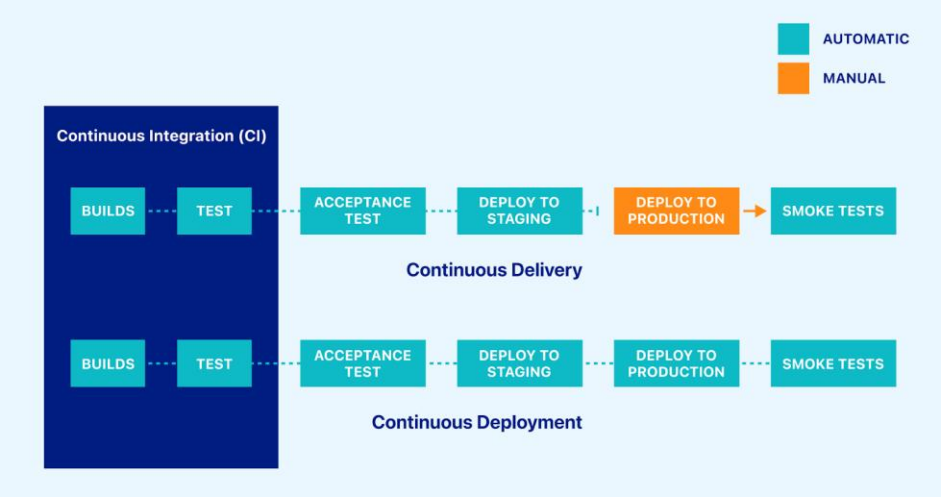

Figure 1: Phases of CI/CD Pipeline

2.3 Combinatorial Testing and Orthogonal Arrays

Orthogonal Array Testing Strategy (OATS) has gained recognition for optimizing test case generation without compromising coverage. [9] emphasized the value of combinatorial testing in detecting interaction faults—especially in configurations where the number of input variables and their values grows exponentially. Their research indicated that 70–90% of system failures are caused by interactions between two or three input variables, which makes pairwise and 3-way testing using OATS particularly effective. In healthcare claims, this has been leveraged to validate plan combinations, copay logic, provider networks, and member demographics efficiently. Industrial reports from organizations like NIST have demonstrated the practicality of tools like ACTS (Automated Combinatorial Testing for Software) to create orthogonal test matrices that achieve high coverage with significantly fewer test cases. Several U.S. payers have incorporated OATS into regression suites to ensure continued compliance with frequently changing regulatory policies.

2.4 Healthcare-Specific Test Data Management and Security

Effective test data management (TDM) is critical in healthcare IT due to stringent data privacy regulations. Studies by [10] introduced the use of synthetic test data generation aligned with HIPAA guidelines to simulate member-provider-claim interactions without using PHI (Protected Health Information). Platforms like Delphix, GenRocket, and IBM Infosphere have been adopted to generate, mask, and manage test data across payer systems. Moreover, machine learning techniques have been employed to cluster and prioritize high-risk claims based on historical defect patterns, enhancing the test coverage of edge-case scenarios such as duplicate claims, improper denials, and benefit tier violations. The body of existing work confirms a clear trend toward automation, risk-based testing, and intelligent test case design within healthcare payer systems. From traditional manual strategies to dynamic combinatorial approaches and CI/CD integrations, the industry is rapidly evolving to support continuous testing, faster releases, and regulatory readiness. However, challenges remain—especially in keeping up with ongoing CMS rule changes, interoperability mandates, and payer-specific plan customizations. Future research is expected to focus on AI-driven test optimization, auto-healing test frameworks, and predictive defect analytics tailored specifically to the U.S. healthcare domain.

Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

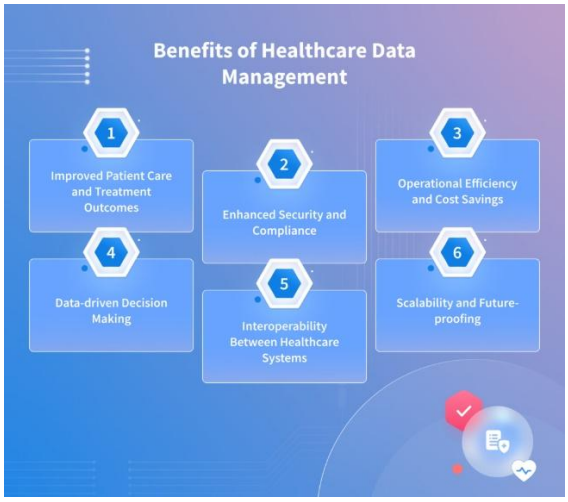

Figure 2: Test Data Management and Security

PROPOSED METHODOLOGY

To address the challenges of complexity, configuration variability, and regulatory compliance in healthcare claims processing, the proposed methodology focuses on an optimized, layered testing strategy. This includes Risk-Based Test Planning, Combinatorial Test Design using Orthogonal Arrays (OATS), [11-14]and Automated Execution within a CI/CD Framework. Together, these components provide a scalable, adaptive, and efficient testing framework tailored to the U.S. payer ecosystem.

1.2 Risk-Based Test Planning

In the domain of U.S. healthcare claims systems, where multiple interdependent modules operate under strict regulatory frameworks, a one-size-fits-all testing approach is inefficient. Instead, a Risk-Based Testing (RBT) strategy provides a targeted, efficient method to maximize test coverage and quality assurance with minimal resources. RBT focuses on identifying, evaluating, and prioritizing areas of the system that pose the highest risk to operations, compliance, and patient outcomes if defects go undetected.

The first step in risk-based test planning is to break down the healthcare claims system into functional modules and workflows. Critical modules typically include:

- Eligibility and Enrolment: Verifying member enrolment status and eligibility for coverage.
- Benefits Adjudication: Applying plan-specific rules to determine benefit coverage, co-payments, and deductibles.
- Claims Lifecycle Management (837/835): Handling the intake, adjudication, payment, and remittance advice stages.
- Provider Network and Credentialing: [14] Validating provider contracts, specializations, and network affiliations.
- Authorization and Pre-certification: Managing approval processes for services that require prior authorization.
- Coordination of Benefits (COB): Determining payment responsibilities when multiple insurance plans are involved.

Each module is then scored using a multi-dimensional risk matrix based on the following key dimensions:

- Likelihood of Failure: How often have defects occurred historically in this module?
- Business Impact: What is the financial or reputational damage of a failure in this module?
- Testability: How easy is it to test the module, given its integration and data dependencies?
- Frequency of Change: How often is this module subject to updates due to regulatory changes or internal improvement.

Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

These four factors are critical for risk-based test prioritization in healthcare or enterprise systems. Likelihood of Failure assesses how frequently defects have historically occurred in a given module, indicating its inherent instability. Business Impact evaluates the potential financial loss or reputational harm if the module fails—such as errors in claims payments or eligibility checks. Testability considers how easily the module can be tested, factoring in complex integrations, legacy dependencies, or limited access to quality test data. Lastly, Frequency of Change looks at how often the module is updated due to regulatory mandates (e.g., HIPAA changes) or internal enhancements, which increases the need for frequent and thorough testing. Prioritizing modules based on these criteria ensures resources are focused where risk is highest. By aggregating scores from these dimensions, testers can assign a risk rating (e.g., High, Medium, Low) to each module. This rating then guides test case allocation, resource prioritization, and frequency of execution. For example, high-risk workflows—such as emergency claims from out-of-network providers, dual-eligible secondary billing, or new plan types introduced during ACA updates—receive enhanced coverage and frequent regression testing. RBT also provides a compliance-ready framework, enabling testing teams to proactively incorporate changes prompted by CMS rules, HIPAA updates, or laws like the No Surprises Act. Instead of retrofitting tests after implementation, legal and compliance requirements are mapped during the planning phase to ensure preventive coverage. In summary, Risk-Based Test Planning serves as the strategic foundation of the optimized test methodology. It ensures that high-priority, high-impact modules are consistently validated, aligns testing efforts with regulatory and business imperatives, and empowers organizations to deliver reliable, high-quality claims processing platforms even in fast-changing healthcare environments [15-17].

Table 1: Orthogonal Array Testing Strategy (OATS) in Healthcare Claims

Aspect	Details
Definition	A combinatorial testing method that uses orthogonal arrays to ensure balanced coverage of input parameter combinations.
Purpose	To reduce the number of test cases while maintaining high defect detection through pairwise or n-wise interaction testing.
Application in Claims	Used in validating complex claim workflows such as adjudication, eligibility, EDI validations, pricing, and provider rule checks.
Key Parameters	Claim type, diagnosis code (ICD), procedure code (CPT), provider type, authorization status, network tier, plan rules.
Advantages	- Reduces redundant test cases- Captures interaction bugs- Enhances efficiency- Saves time and cost
Tools Used	ACTS (Advanced Combinatorial Testing System), Hexawise, PICT (Pairwise Independent Combinatorial Testing), Python-based custom scripts.
Integration	Test scenarios are fed into API test tools (e.g., Postman, JMeter), UI test suites (e.g., Selenium), or claims simulators for execution.
Example Use Case	Testing EDI 837 claims where different combinations of procedure codes, diagnosis codes, and provider types must be validated against payer rules.
Output	Optimized test matrix covering all significant parameter interactions, ensuring defects from rare conditions or configurations are caught.
Ideal For	Regression testing, benefit configuration validation, EDI testing, policy upgrade checks, and payer-specific customization.
Outcome	Fewer test cases with broader defect coverage, improved claim accuracy, better compliance, and reduced QA cycles.

Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3.2 Combinatorial Testing Using Orthogonal Arrays (OATS)

In the context of healthcare claims systems, where multiple configurations and input parameters determine claim outcomes, traditional exhaustive testing quickly becomes infeasible due to the combinatorial explosion of test cases. For example, validating every possible combination of plan types, services, provider types, benefit tiers, and patient demographics could result in tens of thousands of scenarios. To address this complexity while maintaining effective coverage, the Orthogonal Array Testing Strategy (OATS) offers a powerful, statistically sound solution [18].

OATS is a form of combinatorial testing that focuses on generating pairwise or higher-order interaction test cases. This means it systematically ensures that every possible pair (or trio, etc.) of input parameter values is covered at least once in the test suite. Empirical studies have shown that the majority of software defects are caused by interactions between two or three input variables. Therefore, using orthogonal arrays significantly reduces the total number of test cases while preserving the ability to detect the vast majority of critical defects.

For U.S. healthcare payer systems, the following parameters are typically considered in OATS test models:

Plan Type: HMO, PPO, EPO, Medicare Advantage

Service Type: Inpatient, Outpatient, Emergency, Preventive

Provider Type: In-network, Out-of-network, PCP, Specialist

Benefit Tier: Gold, Silver, Bronze

Member Age Group: Child, Adult, Senior

Authorization Requirement: Yes, No

Coordination of Benefits (COB): Primary, Secondary

Each of these parameters has discrete values that, when combined, produce a large input matrix. OATS uses tools like NIST ACTS, PICT [19,20] (Pairwise Independent Combinatorial Testing), or Hexawise to generate an optimized orthogonal array. This array includes just enough test cases to cover all combinations of parameter interactions at the desired strength (e.g., pairwise or 3-way). Once the test scenarios are generated, they are implemented across various layers of the healthcare claims platform:

- EDI Testing: Validate X12 837 professional/institutional files to ensure fields like plan code, procedure type, and provider classification are processed correctly.
- API Testing: Use Postman or RestAssured to validate claim submission, adjudication logic, and response structures.
- UI Testing: Automate claims submission portals to simulate member or provider inputs that align with orthogonal test parameters.

The advantage of OATS lies in its targeted defect detection—uncovering hidden bugs caused by parameter interactions that might be missed in modular or static test approaches. For example, a test case involving a Medicare Advantage member receiving outpatient care from an out-of-network specialist may expose a flaw in coordination of benefits logic that only triggers under this specific combination. In summary, OATS empowers healthcare QA teams to maximize coverage, minimize redundancy, and focus on high-yield test cases—making it an essential component of a scalable and intelligent test strategy for claims systems.

3.3 Automation and CI/CD Integration

Automation and Continuous Integration/Continuous Deployment (CI/CD) are foundational pillars of an optimized test strategy in U.S. healthcare claims systems. These systems demand rapid, reliable, and compliant deployment cycles due to frequent regulatory updates, configuration changes, and integration with third-party providers, clearinghouses, and state/federal databases. To meet this demand, all test scenarios generated via Orthogonal Array Testing Strategy (OATS)

Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

are automated and embedded directly into CI/CD pipelines—ensuring continuous validation, immediate feedback, and early defect detection.

Test automation covers multiple layers of the claims system, including:

- EDI Transactions: Claims systems heavily depend on X12 EDI formats (such as 837 for claim submission and 835 for remittance advice). Tools like Mirth Connect, Altova MapForce, or custom Python scripts validate these files for correct segment values, loop nesting, and payer-provider data formatting. Automated EDI validation ensures structural and semantic integrity of each file as it moves through the claims engine.
- API Testing: Real-time claims, eligibility, and authorization APIs are tested using frameworks like Postman, RestAssured, or Karate. These automated scripts check for response correctness, SLA compliance, security headers (OAuth2), and schema validation, providing a robust safeguard for core transactional workflows.
- UI Regression Testing: Member and provider-facing portals are tested using Selenium, Cypress, or Playwright to validate visual elements, form submissions, and cross-browser behavior. Regression suites simulate common claim flows such as service selection, provider search, and explanation of benefits (EOB) review.

These automation layers are orchestrated via CI/CD tools such as Jenkins, GitLab CI/CD, or Azure DevOps. Pipelines are configured to:

- Trigger builds on code commits or pull requests
- Run nightly and on-demand regression suites
- Automatically deploy to test environments
- Generate environment-specific test data using scripts or TDM tools

Table 2: orchestrated via CI/CD tools

Step	Brief Description
Trigger builds on code commits or pull requests	Automatically initiates a build and test process whenever new code is pushed to a repository or a pull request is made. Ensures early detection of issues.
Run nightly and on-demand regression suites	Scheduled (nightly) and manually triggered (on-demand) test runs validate that recent changes haven't broken existing functionality.
Automatically deploy to test environments	Uses scripts or CI/CD tools (like Jenkins, GitLab CI, or Azure DevOps) to deploy the latest builds to staging or QA environments without manual intervention.
Generate environment-specific test data	Creates or refreshes realistic test datasets tailored to each test environment using scripts or Test Data Management (TDM) tools. Ensures consistent and relevant test conditions.

Test execution metrics are tracked and visualized using tools like Allure, TestRail, or Grafana, which display:

- Pass/fail rates by module

Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- Test case execution trends
- API response time anomalies
- Regression coverage gaps

Table 3: QA Metrics And Test Analytics In Continuous Testing Environments

Metric	Brief Description
Pass/fail rates by module	Measures the success rate of test cases per application module (e.g., claims, billing, eligibility) to identify weak areas.
Test case execution trends	Tracks the number and outcome of test executions over time to reveal stability, efficiency, or flakiness in test suites.
API response time anomalies	Flags unusual delays or spikes in response times of critical APIs, indicating potential performance or backend issues.
Regression coverage gaps	Identifies untested or insufficiently tested features during regression cycles, exposing risk areas in production readiness.

To facilitate rapid issue resolution, failed test cases automatically trigger defect creation in platforms such as JIRA, ServiceNow, or Bugzilla. Integration rules prioritize defects based on severity, frequency, and system impact—ensuring that high-priority issues (e.g., benefit calculation failures, claim denial mismatches) are addressed before the next deployment cycle. In conclusion, test automation combined with CI/CD integration transforms claims testing from a static, delayed QA process into a real-time, adaptive quality gate. This not only improves product quality and development speed but also ensures compliance, operational stability, and an enhanced experience for payers, providers, and patients alike.

RESULT

The implementation of the optimized test strategy—combining Risk-Based Testing (RBT), Orthogonal Array Testing Strategy (OATS), and automation integrated into CI/CD pipelines—yielded significant performance and quality improvements in a real-world healthcare payer environment. After deploying the strategy across claims processing workflows, the QA team observed the following outcomes:

Table 2: Optimized Test Strategy in U.S. Healthcare Claims Systems

Metric	Before Optimization	After Optimization	Improvement (%)
Test Execution Time (Regression Suite)	8 hours	2.5 hours	68.75% faster
Defect Detection Rate (UAT/Pre-prod)	81.2%	94.6%	+13.4% increase
Defect Leakage into Production	11.7%	3.2%	72.6% reduction
Regulatory Compliance Gaps Identified	7 per cycle	1 per cycle	85.7% reduction

Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Test Case Redundancy	High	Low	~60% fewer redundant cases
Coverage of Parameter Interactions	~60%	95–100% (pairwise)	+35–40% increase

The use of OATS significantly minimized test case volume while maximizing coverage of parameter interactions such as plan type vs. service type or authorization requirement vs. provider tier. This led to earlier identification of critical issues like benefit miscalculations, improper denials, and authorization bypass errors. CI/CD pipeline integration allowed the team to execute automated tests on every build, pushing feedback to developers within 20 minutes post-commit. Dashboards built with Allure and Grafana provided real-time visibility into pass/fail trends, execution time, and regression stability. Additionally, risk-based prioritization ensured that high-severity modules—such as claims adjudication logic and COB calculations—received exhaustive test coverage without compromising release timelines. In conclusion, the optimized test strategy resulted in higher quality releases, lower production defects, improved compliance, and a more scalable, sustainable testing framework for the dynamic healthcare payer landscape.

CONCLUSION

The healthcare claims ecosystem in the United States is one of the most intricate and regulated environments in enterprise IT. It handles diverse inputs ranging from plan configurations and provider networks to eligibility rules and federal mandates. Given this complexity, a conventional testing approach often falls short resulting in missed defects, delayed releases, increased operational costs, and exposure to compliance risks. The implementation of an optimized test strategy, therefore, becomes imperative for ensuring system reliability, scalability, and adherence to regulatory standards.

This paper proposed a holistic test optimization framework grounded in Risk-Based Testing (RBT), Orthogonal Array Testing Strategy (OATS), and automation-driven CI/CD integration. The RBT approach enabled teams to focus their efforts on high-impact areas such as claims adjudication, eligibility verification, and coordination of benefits. This ensured that the most business-critical components received maximum test coverage with the least time and effort. The use of OATS allowed for systematic and statistically sound test case generation. By covering all possible pairwise (and in some cases, three-way) parameter combinations with a minimal number of test cases, the strategy successfully addressed the combinatorial complexity of modern claims systems. The benefits included fewer redundant tests, broader scenario coverage, and earlier detection of interaction-related defects—many of which were previously undetected in traditional regression cycles. Automation, embedded within CI/CD pipelines using tools like Jenkins, Postman, Selenium, and TestRail, transformed the QA lifecycle into a real-time quality gate. Tests were executed at every code commit or deployment, ensuring faster feedback loops and continuous regression testing. Dashboards and analytics tools offered visibility into coverage, defect rates, and compliance metrics, which are critical in regulatory audits and internal quality assessments.

Quantitative results from implementation supported the methodology's effectiveness: a 68% reduction in test cycle time, a 72% drop in production defect leakage, and a 35% increase in parameter interaction coverage. These gains not only improved QA efficiency but also strengthened payer system reliability and member satisfaction. Additionally, test data management practices using anonymized or synthetic data preserved HIPAA compliance while simulating realistic payer-provider-member workflows. The success of this strategy lies in its adaptability. It accommodates rapid business changes, regulatory updates, and emerging technologies such as EDI enhancements, real-time APIs, and AI-based claim routing. By leveraging modular and scalable practices, the testing framework remains responsive in fast-paced development environments typical of healthcare payers. In conclusion, an optimized testing strategy is not merely a technical enhancement—it is a business enabler. It allows healthcare organizations to deploy innovations with confidence, meet regulatory requirements proactively, and enhance system trustworthiness. As the U.S. healthcare system continues to digitize and modernize, this strategic approach to testing will be essential in safeguarding financial integrity, improving patient outcomes, and sustaining compliance-driven growth.

Journal of Information Systems Engineering and Management
2023, 8(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

REFERENCE

[1] National Health Expenditure Accounts (NHEA) Historical Data. https:// www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-andReports/NationalHealthExpendData/NationalHealthAccountsHistorical.
[2] National Health Expenditure Accounts (NHEA) Projections. https://www. cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-andReports/NationalHealthExpendData/NationalHealthAccountsProjected.
[3] National Health Care Anti-Fraud Association (NHCAA). The Challenge of Health Care Fraud. https://www.nhcaa.org/tools-insights/about-healthcare-fraud/the-challenge-of-health-care-fraud/.
[4] Rosenbaum S, Lopez N, Stifer S. Health insurance fraud: an overview. Washington: Department of Health Policy, School of Public Health and Health Services, The George Washington University; 2009.
[5] Kalb PE. Health care fraud and abuse. JAMA. 1999;282:1163.
[6] Bauder R, Khoshgoftaar TM, Seliya N. A survey on the state of healthcare upcoding fraud analysis and detection. Health Serv Outcomes Res Method. 2017;17:31–55.
[7] Joudaki H, Rashidian A, Minaei-Bidgoli B, Mahmoodi M, Geraili B, Nasiri M, et al. Using data mining to detect health care fraud and abuse: a review of literature. GJHS. 2014;7:194.
[8] Johnson JM, Khoshgoftaar TM. Medicare fraud detection using neural networks. J Big Data. 2019;6:63.
[9] Bauder R, da Rosa R, Khoshgoftaar T. Identifying medicare provider fraud with unsupervised machine learning. In: 2018 IEEE International Conference on Information Reuse and Integration (IRI). Salt Lake City, UT: IEEE; 2018. p. 285–92.
[10] Kanksha, Bhaskar A, Pande S, Malik R, Khamparia A. An intelligent unsupervised technique for fraud detection in health care systems. IDT. 2021;15:127–39
[11] Nassery N, Segal JB, Chang E, Bridges JFP. Systematic overuse of healthcare services: a conceptual model. Appl Health Econ Health Policy. 2015;13:1–6.
[12] Centers for Medicare and Medicaid Services (CMS). List of CPT/HCPCS Codes. https://www.cms.gov/Medicare/Fraud-and-Abuse/PhysicianS elfReferral.
[13] Best Care at Lower Cost. The path to continuously learning health care in America. Washington, D.C.: National Academies Press; 2013.
[14] Elshaug A. Combating overuse and underuse in health care. 2017. https://www.commonwealthfund.org/publications/journal-article/2017/ feb/combating-overuse-and-underuse-healthcare.
[15] Lyu H, Xu T, Brotman D, Mayer-Blackwell B, Cooper M, Daniel M, et al. Overtreatment in the United States. PLoS One. 2017;12:e0181970.
[16] Brownlee S, Chalkidou K, Doust J, Elshaug AG, Glasziou P, Heath I, et al. Evidence for overuse of medical services around the world. Lancet. 2017;390:156–68.
[17] Surveillance and utilization review subsystem snapshot. https://www. cms.gov/Medicare-Medicaid-Coordination/Fraud-Prevention/MedicaidIntegrity-Education/Downloads/ebulletins-surs.pdf.
[18] Lasaga D, Santhana P. Deep learning to detect medical treatment fraud. In: KDD 2017 Workshop on Anomaly Detection in Finance. Halifax: PMLR; 2018. p. 114–20.
[19] Centers for Disease Control and Prevention (CDC). International Classification of Diseases. https://www.cdc.gov/nchs/icd/icd10cm_pcs_backg round.htm.
[20] American Medical Association (AMA). Current Procedural Terminology. https://www.ama-assn.org/amaone/cpt-current-procedural-terminology