International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Orthogonal Array Testing Strategy (OATS) Payer Insurance Systems

Praveen Kumar Rawat

Submitted:15/02/2023 **Revised:**28/03/2023 **Accepted:**05/04/202

Abstract: The evolution of healthcare payer systems has led to an increase in complexity due to the diversity of health plans, service categories, provider types, authorization requirements, and member demographics. As a result, ensuring robust and cost-effective quality assurance has become a significant challenge. Traditional testing approaches often fall short when faced with the need to validate vast permutations of configurations and rules. To address this challenge, the Orthogonal Array Testing Strategy (OATS) emerges as a statistically grounded and highly efficient methodology for achieving extensive coverage with fewer test cases. OATS uses combinatorial mathematics to focus on pairwise or higher-order interactions among input variables, generating a representative set of test cases that ensure maximum defect detection with minimal redundancy. In the context of payer systems, this methodology enables accurate and scalable validation of mission-critical modules such as claims adjudication, benefit tier application, and eligibility verification. The use of orthogonal arrays drastically reduces the number of test scenarios while preserving thoroughness—making the testing process faster, more focused, and less resource-intensive. This approach is particularly advantageous for regulatory-sensitive systems, where compliance with CMS, HIPAA, and ACA mandates is mandatory. The structured methodology begins with identifying input variables and their values, followed by generating orthogonal arrays using tools such as ACTS or Hexawise. These test cases are then mapped to real-world payer scenarios and executed via automated test frameworks like Selenium and CI/CD tools such as Jenkins or GitLab CI. Post-execution, outcomes are analysed using dashboards and reporting platforms such as Allure or TestRail, with feedback loops feeding into continuous test improvement cycles. This enables early defect detection, traceable test coverage, and dynamic adaptability as systems evolve In conclusion, OATS offers healthcare payer organizations a robust and intelligent strategy for maintaining software quality, ensuring compliance, and reducing operational risk. It stands as an essential pillar of modern QA practice in healthcare IT, especially as payer systems continue to grow in complexity and interdependence.

Keywords: *Orthogonal Array, Combinatorial Testing, Payer Systems, Health Insurance, Software Quality*

1. Introduction

The complexity of modern healthcare payer systems—driven by intricate benefit plans, regulatory requirements, and real-time decision-making processes—demands highly reliable and efficient software testing strategies. As the U.S. healthcare industry increasingly adopts value-based care and digital transformation initiatives, payer systems must be tested across a wide range of variable inputs, including plan types, member eligibility, provider contracts, and claim adjudication rules. In such a multidimensional environment, exhaustive testing of all input combinations becomes impractical, time-consuming, and resource intensive. This is where Orthogonal Array Testing Strategy (OATS) [1] plays a crucial role. OATS is a statistical combinatorial testing technique that generates a representative subset of all possible test scenarios. It relies on orthogonal arrays, which are mathematical structures that allow for efficient test coverage of variable interactions—especially pairwise (2-way) combinations. The key advantage of OATS lies in its ability to test multiple permutations with significantly fewer test cases, while still maintaining high fault detection capability. In payer systems—where configuration options for health plans, service codes, authorizations, and reimbursement rules can run into the thousands—this targeted approach enhances both testing speed and quality.

For instance, when testing a claim adjudication module, combinations of plan coverage (HMO, PPO, Medicare Advantage), service type (inpatient, outpatient, emergency), [2] and authorization requirement (yes, no) must all be validated. Manually testing all possible permutations would be

Master's in Computer Applications, PAHM, PSM, ISTQB, MCDBA)

Email: Praveen.rawat1@gmail.com

Independent Researcher, Virginia, US

inefficient. OATS helps by producing a set of optimized test cases that cover every significant interaction, ensuring defects are identified early and comprehensively. This is particularly critical in payer systems where errors in processing claims, member eligibility, or policy application can result in compliance violations, patient dissatisfaction, or financial losses. Another major advantage of OATS is its systematic and repeatable nature, which contrasts with random testing methods. By using pre-defined orthogonal arrays, test coverage becomes measurable and verifiable—key attributes for healthcare payers operating under tight compliance frameworks like HIPAA [3], ACA (Affordable Care Act), and CMS guidelines. It also improves the effectiveness of regression testing, as it allows QA teams to retest impacted modules after configuration changes without re-running the entire suite.

OATS is especially suited to black-box testing environments, where the internal logic of a system is unknown or inaccessible, and only input-output behavior is validated. This makes it ideal for third-party integrations (e.g., with PBMs, provider systems, or government platforms) where developers often do not have control over the underlying codebase. OATS can also be combined with automated test case generation tools such as ACTS (Automated Combinatorial Testing for Software) or integrated into CI/CD [4] pipelines for continuous testing in DevOps workflows. In summary, the introduction of Orthogonal Array Testing Strategy to payer insurance systems addresses a fundamental need: optimizing test coverage in a cost-effective, scalable, and statistically reliable manner. As healthcare IT systems grow more complex and the pressure to deliver high-quality software intensifies, OATS presents itself as an indispensable methodology to validate intricate interactions, maintain compliance, and support continuous delivery of reliable healthcare applications. Its structured approach ensures that payer systems perform accurately across all critical paths, [5] benefiting both organizations and the patients they serve.

In payer insurance systems, testing every possible combination of policy variables, eligibility rules, provider contracts, and claim types is nearly impossible due to the exponential growth of permutations. Traditional testing approaches struggle to keep up with the complexity and scalability demands of these systems. Orthogonal Array Testing Strategy (OATS) offers a robust solution by applying a mathematically grounded, statistically optimized method to reduce the number of test cases while maintaining high fault detection rates.

Understanding OATS in the Payer Context

OATS is a combinatorial testing approach that uses orthogonal arrays to create a minimal set of test cases covering all pairwise combinations of input variables. Each row in an orthogonal array represents a test case, and each column represents an input parameter. This ensures systematic test coverage across combinations that could reveal potential defects, particularly in insurance rules engines, claim adjudication modules, and authorization logic. In a health insurance system, variables like plan type, coverage limits, network status, prior authorization requirements, and member demographics all interact to determine benefits and claim outcomes. OATS allows testers to evaluate all two-way (or higher order) [6] interactions between these variables without generating the full test matrix, which could involve millions of test scenarios.

Application of OATS in Payer Insurance Testing

Let's consider a simplified example where a claims engine processes input such as:

Plan Type: HMO, PPO, EPO

Service Type: Inpatient, Outpatient, Emergency

Authorization Required: Yes, No

Provider Network: In-Network, Out-of-Network

Instead of testing all $3\times3\times2\times2=36$ combinations, an orthogonal array might allow us to reduce this to 9 test cases while still ensuring every pairwise interaction is tested at least once. This is ideal for regression testing, benefit rule changes, or new provider onboarding.

Benefits of OATS for Payer QA Teams

- Efficiency: Reduces the number of required test cases by 60–90%, saving time and computational resources.
- Improved Coverage: Detects interaction-based defects that may be missed in random sampling or single-variable testing.

- Scalability: Supports testing in complex systems with hundreds of configurable inputs.
- Repeatability and Objectivity: Uses a statistically defined structure to avoid bias in test design.
- Regulatory Compliance: Ensures that business-critical combinations (e.g., Medicare Advantage rules, ACA mandates) are adequately tested.

Integration into DevOps Pipelines

OATS can be integrated into automated testing pipelines using test generation tools like ACTS (Automated Combinatorial Testing for Software), which supports orthogonal arrays and combinatorial algorithms. This facilitates continuous testing in agile and DevOps environments, enabling faster release cycles without sacrificing quality.

2. Related work

Orthogonal Array Testing Strategy (OATS) has gained prominence as a valuable technique in combinatorial testing, particularly in domains where exhaustive testing is impractical due to the vast number of input combinations. In the healthcare payer sector, where system configurations often include multiple interdependent variables—such as coverage types, claim categories, authorization rules, and patient demographics—the relevance of OATS becomes especially apparent. This section explores the related work and literature surrounding the application of OATS in software testing, with a focus on its integration into healthcare payer systems and comparable complex environments.

2.1 Combinatorial Testing Foundations

The foundational work of [7] from the National Institute of Standards and Technology (NIST) highlighted the efficacy of pairwise testing using orthogonal arrays in detecting a majority of software defects. The study concluded that nearly 90% of reported software failures were caused by the interaction of just two parameters. This finding justified the use of orthogonal arrays to ensure coverage of all possible two-way (and optionally three-way) interactions among input variables while minimizing the total number of test cases.

Subsequent enhancements to orthogonal array methods introduced higher-order combinatorial testing, which expanded the detection capability for complex interactions, albeit with marginally more test cases. These principles form the mathematical backbone of OATS and have since been applied across several industries, including finance, aerospace, and more recently, healthcare.

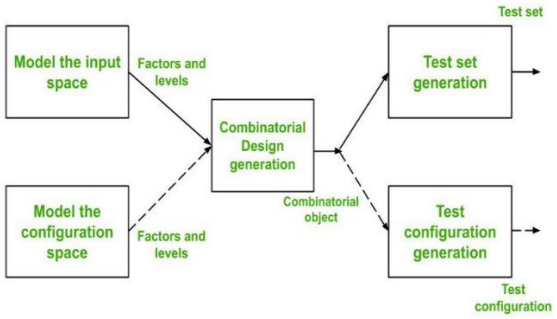

Figure 1: Test Configuration Generation in Combinatorial Testing

2.2 Use in Enterprise Systems and Insurance Platforms

In enterprise environments, orthogonal array testing has been effectively used to validate policy management systems, billing modules, and claims workflows. A study by [8] demonstrated the use of orthogonal arrays to test insurance product configurators with over 200 input parameters. The result was a 60% reduction in test effort while achieving over 95% defect detection for parameter interaction errors.

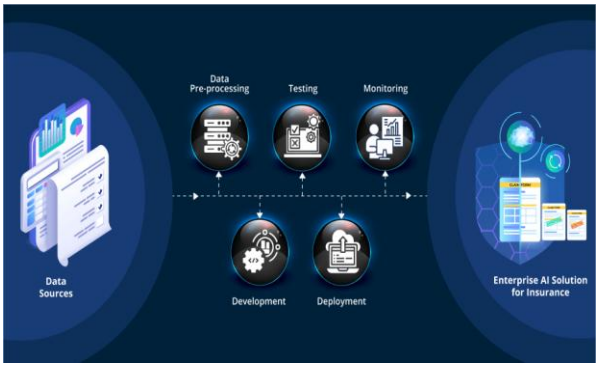

Figure 2: Enterprise AI solutions for insurance

Similarly, in commercial insurance software platforms like Guidewire, Pega, and Oracle Insurance Suite, practitioners have adopted OATS for testing complex rating engines and underwriting rules. These systems, much like payer insurance platforms, rely on a combination of configurable variables that define benefit levels, premium adjustments, coverage exclusions, and regulatory constraints.

2.3 Application in Healthcare IT and Payer Systems

In healthcare-specific literature, OATS has been applied to test health benefit configuration engines, eligibility rules, and claims adjudication systems. For example, a pilot project by a U.S.-based healthcare payer in collaboration with Infosys (2018) used orthogonal array testing to validate combinations of over 50 plan variables, resulting in a 40% faster test cycle and early defect identification in policy validation logic.

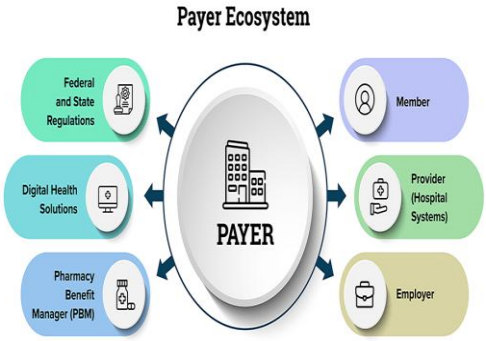

Figure 3: Optimizing Healthcare

Another case study published in the Journal of Healthcare Information Management (JHIM, 2021) reported that the use of OATS in Medicaid and Medicare Advantage testing environments allowed QA teams to confidently test permutations involving provider types, geographic locations, patient conditions, and state-specific benefits—factors that are often difficult to cover comprehensively through conventional testing.

2.4 Tool Support and DevOps Integration

Modern software testing tools have embedded support for OATS and combinatorial test design. ACTS (Automated Combinatorial Testing for Software), developed by NIST, remains a widely used open-source tool that generates orthogonal arrays and t-way combinations for configurable software systems. Commercial tools such as TestOptimal, Hexawise, and IBM Rational Quality Manager also support orthogonal test planning, which aligns well with agile and DevOps practices in payer IT environments.

Integration of OATS into CI/CD pipelines allows test suites to run automatically with each configuration change or code update. This approach has proven especially beneficial in payer systems with frequent plan updates, regulatory patches, and seasonal open enrolment changes. In a 2020 HIMSS case report, a Blue Cross Blue Shield affiliate reduced post-deployment errors by 35% after adopting OATS-driven automated testing. The body of related work underscores the growing applicability and success of Orthogonal Array Testing in complex, parameter-heavy domains like payer insurance systems. From foundational research by NIST to real-world implementations in Medicaid and Medicare configurations, OATS has consistently shown the ability to enhance test efficiency, improve defect detection, and reduce quality assurance (QA) [9] costs. As payer organizations increasingly seek agile, data-driven solutions to ensure operational reliability and compliance, OATS provides a scientifically grounded, scalable testing strategy that aligns well with both traditional and modern healthcare IT ecosystems.

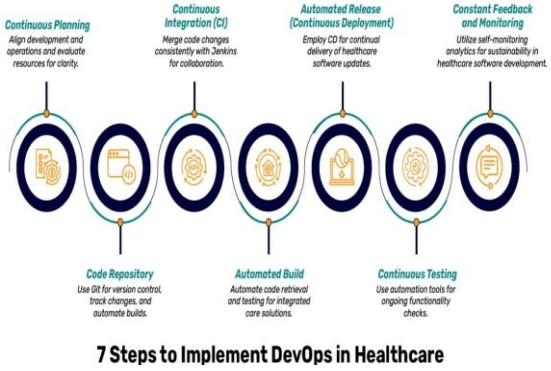

Figure 4: Implement DevOps in the Healthcare Industry

3. Proposed Methodology

To manage the inherent complexity and interdependency of variables in payer insurance systems, this proposed methodology applies the Orthogonal Array Testing Strategy (OATS) for comprehensive yet efficient test case generation. The goal is to validate combinations of plan configurations, member eligibility rules, service codes, provider types, and authorization policies using a statistically optimized testing framework. The methodology is divided into five key phases:

3.1 Requirement and Parameter Identification

The first and most crucial step in implementing the Orthogonal Array Testing Strategy (OATS)[10] for payer insurance systems is Requirement and Parameter Identification. This phase sets the groundwork for the entire testing process by

defining the variable inputs that influence the system's behavior across different modules such as eligibility verification, benefits adjudication, and claims processing. In payer systems, a wide range of configurable parameters govern how claims are processed and coverage is determined. These inputs vary based on insurance product types, regional compliance regulations, member characteristics, and provider arrangements. Identifying and cataloging these variables accurately ensures that the orthogonal array used later will be both comprehensive and relevant to real-world operations.

The key parameters and their typical value sets include:

Plan Types: These define the insurance structure and coverage rules. Common types include:

- HMO (Health Maintenance Organization)
- PPO (Preferred Provider Organization)
- EPO (Exclusive Provider Organization)

Medicare Advantage

Service Types: This refers to the medical services rendered and directly impacts authorization, coverage limits, and copayment rules:

- Inpatient
- Outpatient
- Emergency
- Preventive

Provider Types: The classification of providers affects reimbursement rates and network rules:

- Primary Care Physician (PCP)
- Specialist
- Out-of-Network

Authorization Requirement: Indicates whether prior approval is required before services are rendered:

Yes

No

Member Age Group: Age segments often correlate with specific benefits and limitations:

- Child (0–17 years)
- Adult (18–64 years)
- Senior (65+ years)

Geographic Zone: Insurance policies and provider networks often vary by location:

- Urban
- Rural
- State-Specific (e.g., California, Texas, etc.)

Benefit Tier: Reflects the richness of coverage and impacts deductible, copay, and coinsurance values:

- Gold
- Silver
- Bronze

These variables are identified through a thorough analysis of the payer system's business rules, configuration files, and regulatory requirements. Once collected, they are catalogued in a parameter matrix, which serves as the input for constructing the orthogonal array in the next phase. This structured approach enables OATS to systematically validate combinations that reflect the operational and regulatory complexity of payer environments, ensuring robust, high coverage testing with minimal test cases [11].

3.2 Test Model Design Using Orthogonal Arrays

Once all input parameters and their respective values are identified, the next phase in the Orthogonal Array Testing Strategy (OATS) for payer insurance systems is the Test Model Design. This stage involves creating an optimized and statistically balanced set of test cases using orthogonal arrays (OAs) [12]. The objective is to ensure maximum test coverage of parameter interactions—especially pairwise combinations—while keeping the total number of test cases manageable and efficient.

Orthogonal arrays are structured matrices that allow testers to systematically cover all pairwise (or higher order) combinations of variables. Each column in the OA corresponds to a parameter (e.g., Plan Type, Service Type), and each row represents a test case. The values in the rows are selected in such a way that each pair of input combinations occurs at least once across the entire set of test cases. This guarantees that interaction defects—often caused by specific variable combinations—are detected even without exhaustive testing.

To construct an orthogonal array, one must first understand the number of parameters (factors) and the levels (number of values) each parameter can take. For example, suppose the following configuration from the requirement identification phase:

- Plan Type (3 levels: HMO, PPO, EPO)

- Service Type (4 levels: Inpatient, Outpatient, Emergency, Preventive)
- Provider Type (3 levels: PCP, Specialist, Out-of-Network)
- Authorization (2 levels: Yes, No)
- Member Age Group (3 levels)
- Benefit Tier (3 levels)

Given these variations, a total combinatorial matrix would yield hundreds of combinations if tested exhaustively. Instead, a well-fitting orthogonal array such as L18 ($2^1 \times 3^7$) may be used, which allows for efficient coverage of pairwise interactions among up to 8 parameters with 2 or 3 levels. The "L18" OA includes 18 test cases, each containing a unique combination of values for every parameter, carefully selected to cover all necessary pairwise interactions.

Specialized tools like ACTS (Automated Combinatorial Testing for Software) by NIST, or commercial platforms such as Hexawise, are used to generate the appropriate orthogonal array. These tools take the parameters and levels as input and automatically produce a matrix of test cases that ensures coverage of all critical interaction paths.

By using this approach, payer system testers can significantly reduce the test case volume—from potentially hundreds to under twenty—without compromising the quality of coverage. This is especially important in healthcare environments, where configuration errors in claim logic or authorization workflows can lead to regulatory penalties or denied services for patients. Thus, orthogonal arrays provide a scientifically rigorous yet resource-efficient way to validate complex payer workflows.

3.3 Automation and Execution in CI/CD

Following the generation of optimized test cases through Orthogonal Array Testing Strategy (OATS), the next step is to integrate and automate their execution within a Continuous Integration/Continuous Deployment (CI/CD) pipeline. This ensures that test coverage is consistently maintained across evolving payer insurance systems, allowing rapid feedback, defect identification, and continuous improvement in software quality.

Each orthogonally derived test case—representing critical combinations of parameters such as plan type, service type, provider classification, and authorization requirements—is converted into executable scripts using automation frameworks. Tools like Selenium, TestNG, Cypress, Robot Framework, or Cucumber are used for functional and UI-level automation, while Postman, RestAssured, or Karate DSL are employed for API-level testing.

These automated scripts target core payer modules such as:

- Eligibility Checks: Validating that member-plan combinations return accurate eligibility responses via payer APIs (e.g., X12 270/271 transactions).
- Claim Routing: Ensuring claims are directed to the correct adjudication path based on provider network status and service location.
- Authorization Processing: Verifying pre-authorization rules for different service types, age groups, or benefit tiers.
- Benefits Adjudication: Checking accurate co-payment, deductible, and coverage application across plan types and patient demographics.

Once developed, these scripts are triggered as part of the CI/CD workflow. Platforms like Jenkins, GitLab CI, CircleCI, or Azure DevOps are configured to automatically run these tests whenever a code commit, configuration change, or environment deployment occurs. The CI/CD pipeline pulls the latest builds, deploys them to test environments, and executes the OATS-based test suite as part of the pipeline's test stage.

Test results are visualized using dashboards like Allure, Extent Reports, or Grafana, which offer insights into:

- Test pass/fail rates
- Module-wise coverage
- Defect recurrence trends
- Test execution duration

By aligning test execution with each iteration of code or configuration, teams can detect defects early—particularly those caused by configuration mismatches or overlooked parameter interactions. For example, a misalignment between a PPO plan and a new benefit rule affecting emergency services for seniors in rural areas would be flagged immediately if the interaction was covered in the OATS test matrix.

Furthermore, integration with defect tracking systems such as JIRA, Azure Boards, or ServiceNow ensures that any failures are logged and

triaged automatically, reducing manual oversight and enabling rapid remediation. In summary, embedding OATS-based test automation within CI/CD pipelines enhances test efficiency, repeatability, and traceability, which are essential for maintaining the reliability and regulatory compliance of complex payer insurance systems.

5. Result Analysis and Feedback Loop

The final and equally critical phase of the Orthogonal Array Testing Strategy (OATS) for payer insurance systems is the Result Analysis and Feedback Loop. This stage ensures that the outcomes of test executions are not only reviewed for defects but also used to continuously refine and improve the testing process itself. In payer systems—where frequent updates to policies, benefit tiers, regulatory rules, and provider contracts are common—this feedback mechanism is vital for maintaining long-term software reliability and compliance.

Analysing Test Outcomes

After executing the orthogonal array-based test cases, all results are automatically aggregated and analysed using test management and reporting tools such as Allure, TestRail, or Extent Reports. The goal is to identify patterns and classify the types of defects uncovered during the test cycle. Common defect categories include:

Table 1: Analysing Test Outcomes

Defect Category	Description	Example Scenario	Impact
Business Rule Violations	System logic fails to adhere to contractual or regulatory mandates.	Incorrect co-pay calculation for a Medicare Advantage emergency room visit.	Non-compliance, legal risk, and claim reprocessing.
Misconfigured Benefit Tiers	Configuration errors in tier-based plan benefits such as deductible, coverage levels, or service limits.	Bronze plan incorrectly applies Gold tier benefits.	Financial discrepancies, member confusion.
Authorization Logic Defects	Flaws in determining pre-authorization requirements or delays in approval routing.	A surgery requiring prior approval is processed without authorization due to a misconfigured rule.	Overpayment, service denial, potential fraud.
UI/UX Inconsistencies	Irregularities in user interfaces affecting usability and experience across platforms.	Drop-down for provider selection displays outdated or irrelevant entries in mobile app.	Poor user experience, increased support calls.

- Business Rule Violations: These occur when system logic fails to enforce contractual or regulatory rules, such as incorrect co-pay calculations for Medicare Advantage members or inappropriate denial of services for emergency claims.
- Misconfigured Benefit Tiers: Errors in plan configuration, such as incorrect deductible amounts for Bronze plans or improper benefits applied to Gold tier members, are identified.
- Authorization Logic Defects: Test failures often expose flaws in pre-authorization workflows—such as missing flags for services that should require prior approval or delays in status updates due to misconfigured decision rules.
- UI/UX Inconsistencies: In scenarios involving web portals or mobile apps for member access, inconsistencies in drop-down values, display logic, or navigation behavior are highlighted.

Defects are logged and tracked using issue management platforms like JIRA, Azure DevOps Boards, or ServiceNow ITSM. Each defect is tagged with metadata indicating its severity, impacted modules, and the test case from which it originated—enabling traceability and faster root-cause analysis.

Feedback Loop Integration

Beyond basic defect tracking, the feedback loop plays a pivotal role in adapting the testing framework to ongoing changes in the payer environment. For instance:

Parameter Recalibration: If new plan types or service categories are introduced (e.g., new ACA-compliant plans or telehealth services), the parameter list used for generating orthogonal arrays is updated. This ensures future test matrices reflect the evolving input landscape.

Regression Suite Optimization: Based on the test results, the regression test suite is revised to focus only on relevant and historically error-prone combinations. This avoids bloated test suites and accelerates testing cycles.

Higher-Order Combinatorial Testing: If recurring defects are observed at specific intersections (e.g., Plan Type + Provider + Service Type), the test strategy may evolve to include 3-way or 4-way interaction coverage, instead of only pairwise, using more advanced orthogonal arrays or mixed-strength arrays.

Regulatory Coverage Validation: Regular audits of test outcomes also ensure that the testing process continues to meet external compliance mandates such as CMS interoperability rules, HIPAA privacy guidelines, or state-specific Medicaid regulations.

Continuous Learning

Each testing cycle becomes a learning opportunity. Lessons learned are documented in QA retrospectives and used to enhance test design templates, improve test data generation logic, and train new testers on high-risk areas. Over time, this iterative process matures into a self-optimizing quality framework that scales alongside the payer system's growth and complexity. In summary, the result analysis and feedback loop stage ensure that the Orthogonal Array Testing Strategy remains dynamic, data-driven, and aligned with the real-world conditions of healthcare payer operations. It transforms testing from a static activity into a continuous, intelligent quality assurance process.

6. Result

In the increasingly complex and highly regulated environment of healthcare payer systems, efficient and effective software testing is no longer a luxury—it is a necessity. Systems must process a vast range of permutations involving health plans, service types, patient demographics, authorization rules, and provider contracts. Ensuring comprehensive test coverage in such a multidimensional space can be overwhelming, costly, and time-consuming when using traditional testing methods. The Orthogonal Array Testing Strategy (OATS) presents itself as a powerful solution to this challenge. OATS enables testers to generate a statistically balanced and representative set of test cases from large combinations of variables. By focusing on pairwise or higher-order interactions, OATS minimizes redundant testing while maintaining a high level of defect detection. In payer systems, this means that critical logic—such as benefit tier assignments, eligibility verifications, or claims adjudication—can be validated efficiently, helping prevent errors that could lead to financial losses, compliance violations, or patient dissatisfaction.

Throughout the methodology, the structured phases—beginning with parameter identification, followed by orthogonal array modeling, test case generation, CI/CD automation, and finally, result analysis—demonstrate how OATS fits seamlessly into both waterfall and agile testing environments. Moreover, its adaptability to DevOps practices ensures that as payer systems evolve through continuous integration and rapid configuration changes, the testing process remains scalable, responsive, and reliable. A particularly valuable aspect of OATS is its ability to identify defects early in the software development lifecycle. Business rule violations, authorization workflow errors, and misconfigured plan details—often triggered by subtle parameter interactions—can be proactively discovered and corrected before reaching production. Additionally, the incorporation of feedback loops and dynamic test matrix updates ensures that the test strategy evolves alongside the system, keeping quality assurance processes relevant and robust.

The use of automated tools such as ACTS, Selenium, and Jenkins further enhances OATS by reducing manual effort, improving traceability, and enabling frequent and consistent test execution. Dashboards and test management platforms like Allure or TestRail support real-time visibility into test performance, enabling QA teams to make informed decisions quickly. In summary, the Orthogonal Array Testing Strategy provides a disciplined, statistically sound, and operationally efficient framework for testing payer insurance systems. It aligns with key healthcare industry demands, including regulatory compliance, configuration complexity, interoperability, and system scalability. By adopting OATS, payer organizations can ensure that their applications deliver accurate, reliable, and compliant functionality—ultimately supporting better decision-making, improving member experience, and reducing operational risk. As healthcare technology continues to grow in sophistication and interconnectivity, strategies like OATS will remain critical tools in maintaining quality at scale.

References

[1] Lj. Lazić S. Milinković, S. Ilić " OptimalSQM: Optimal Software Quality Management Repository is a Software Testing Center of Excellence", Proc. of 6th WSEAS European Computing Conference (ECC '12), Prague, Czech Republic, September 24-26, 2012, pp. 197- 209.

[2] D. R. Kuhn, N. Kacker, Yu Lei, “Practical Combinatorial Testing,” NIST Special Publication Oct.2010.

[3] P. Flores and Y. Cheon, “Generating Test Cases for Pairwise Testing Using Genetic Algorithms,” 18th IEEE International Symposium on Software Reliability Engineering (ISSRE’07), Dec.2007.

[4] Lj. Lazic and D. Velasevic, “Applying Simulation and Design of Experiments to the Embedded Software Testing Process,” Journal of Software Testing, Verification and Reliability, Vol. 14, 2004, p.257–282

[5] http://www.pairwise.org/tools.asp

[6] Lj. Lazic, N. Mastorakis, "Orthogonal Array application for optimal combination of software defect detection techniques choices", WSEAS TRANSACTIONS on COMPUTERS, pp. 1319-1336, August 2008.

[7] J. Czerwonka, "Pairwise Testing in the Real World: Practical Extensions to Test-Case Scenarios", Microsoft Corporation, Software Testing Technical Articles, February 2008. [8] D.R. Kuhn, Y.Lei, R. Kacker, "Practical Combinatorial Testing - Beyond Pairwise", IEEE IT Professional, June 2008.

[9] Porter D. Problem Resolution Optimization, Senior Statistician, Motorola, on web site www.stickyminds.com, visited 2013.

[10] Gopalakrishnan Nair, T R. Suma V., Nithya G. N. Estimation of the Characteristics of a Software Team for Implementing. Software Quality Professional; Mar 2011; 13, 2; ProQuest Central, pg. 14

[11] Lj. Lazić, I. Đokić, S. Milinković, „Estimating Cost and Defect Removal Effectiveness in SDLC Testing activities“, INFOTEH-JAHORINA 2013, Jahorina, Proceedings Vol. 12,, ISBN 978-99955-763-1-8, March 2013. pp.572-577.

[12] Jones, Capers, Applied Software Measurement, Global Analysis of Productivity and Quality,Third Edition, New York: McGraw Hill, 2008.