Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Practical Framework for Migrating Large Manual Test Suites to Automation Pipelines: An Industrial Case Study

Dimple Bajaj[1], Vishal Shah[2], Sandeep Kanaparthi[3]

[1]Sr. Supervisor, Software Quality Engineering

[2]Senior engineering manager

[3]Principal Software Engineer

ARTICLEINFO

ABSTRACT

Enterprise software systems frequently accumulate large manual regression test suites that become costly and inefficient to maintain. While test automation is widely recognized as a mechanism for improving quality and release velocity, organizations often struggle to migrate legacy manual test repositories into sustainable automation pipelines. This paper presents the Manual Test Automation Framework (MTAF), an industrial experience-driven and architecture-aligned methodology for systematically transforming large manual regression inventories into scalable automation ecosystems. The framework emphasizes architectural traceability, structured test inventory normalization, risk-based prioritization, tool capability evaluation, and layer-aligned automation strategy mapping. MTAF was applied to a cloud-deployed enterprise web application supporting multi-role workflows and distributed deployments. The transformation reduced the regression suite from 784 to 401 test cases, automated 180 high-impact scenarios, and reduced regression execution time from approximately 12 days to 3 days.This work contributes a transferable industrial framework that organizations can adopt to rationalize legacy test repositories and implement sustainable automation strategies without disrupting business continuity.

Keywords: Software Test Automation, Regression Testing, Test Migration, Software Quality Engineering, Industrial Case Study

1 INTRODUCTION

Regression testing plays a critical role in maintaining software quality as systems evolve. However, in many long-lived enterprise applications, regression validation remains largely manual. Over time, manual test repositories grow organically, often without systematic review or pruning, resulting in redundant, obsolete, or low-value test cases. As a consequence, regression cycles become increasingly expensive and time-consuming.

Despite the availability of numerous automation frameworks and tools, organizations frequently struggle to adopt test automation successfully. Prior studies indicate that automation initiatives frequently encounter challenges due to unclear prioritization, insufficient system understanding, inappropriate tool selection, and excessive reliance on UI-centric automation.

Existing research primarily focuses on automation frameworks, model-based testing, and test generation techniques. In contrast, there is limited guidance on how to practically migrate large legacy manual test inventories into sustainable automation pipelines while preserving business

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

continuity. This paper addresses this gap by presenting a framework derived from direct industrial experience.

While regression prioritization and automation strategies have been extensively studied in software engineering research, practical guidance on migrating large legacy manual repositories into sustainable automation pipelines remains limited.

This work addresses that gap by proposing and evaluating a phased migration framework grounded in architectural traceability, structured inventory rationalization, and risk-aligned automation sequencing.

Unlike tool-centric automation approaches that begin with script development or framework selection, MTAF prioritizes architectural mapping and repository normalization before automation execution.

The Primary Contributions Of This Paper Are As Follows.

- A phased framework for migrating large manual regression suites to automation pipelines grounded in architectural traceability.
- An experience-based approach for test inventory normalization and relevance filtering prior to automation implementation.
- A risk-driven prioritization model to sequence automation adoption and demonstrate early return on investment.
- A hybrid, layer-aligned automation strategy for layered enterprise systems combining UI, API, and backend validation.
- Empirical industrial validation demonstrating measurable efficiency gains and transferability potential.

Methodology

This paper is positioned as an industrial experience report and framework proposal rather than a controlled experimental study. The objective is to formalize a transferable migration methodology derived from practical execution in a large-scale enterprise environment. The contribution lies in structuring the migration process into reproducible phases grounded in architectural alignment, risk-driven sequencing, and repository rationalization.

2 RELATED WORK

Regression test optimization has been widely studied in software engineering literature. Rothermel and Harrold [7] introduced foundational regression test selection techniques aimed at reducing execution cost while preserving fault detection effectiveness. Subsequent surveys by Yoo and Harman [8] synthesized regression test minimization, selection, and prioritization strategies across algorithmic and heuristic approaches. Empirical studies on test case prioritization have further demonstrated measurable fault detection improvements [5].

Garousi and Mäntylä [1] conducted a multivocal literature review of industrial test automation practices, emphasizing that automation success depends on systematic planning rather than tooling alone.

Automation sustainability in continuous delivery environments has also been examined in DevOps research highlighting pipeline integration challenges [11]. Despite extensive research on prioritization, model-based testing [2], and automation tooling [6], comparatively less attention has been given to structured migration of large legacy manual repositories into layered automation strategies grounded in architectural traceability and inventory rationalization.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3 SYSTEM CONTEXT

The Manual Test Automation Framework was implemented in an enterprise web-based training and certification platform that supports multiple user roles and workflow-driven interactions. The system follows a layered web application architecture consisting of presentation, service, API, asynchronous processing, and data persistence layers.

The application is deployed on cloud infrastructure across multiple geographic regions, including region-specific instances. The system integrates with multiple external enterprise services and includes both synchronous request-response workflows and asynchronous background processing.

Prior to the transformation, regression validation relied on a manual test suite, typically executed during major releases, requiring approximately 12 days to complete. Test cases were selected primarily based on perceived impact of change and did not provide comprehensive regression coverage due to resource and schedule constraints.

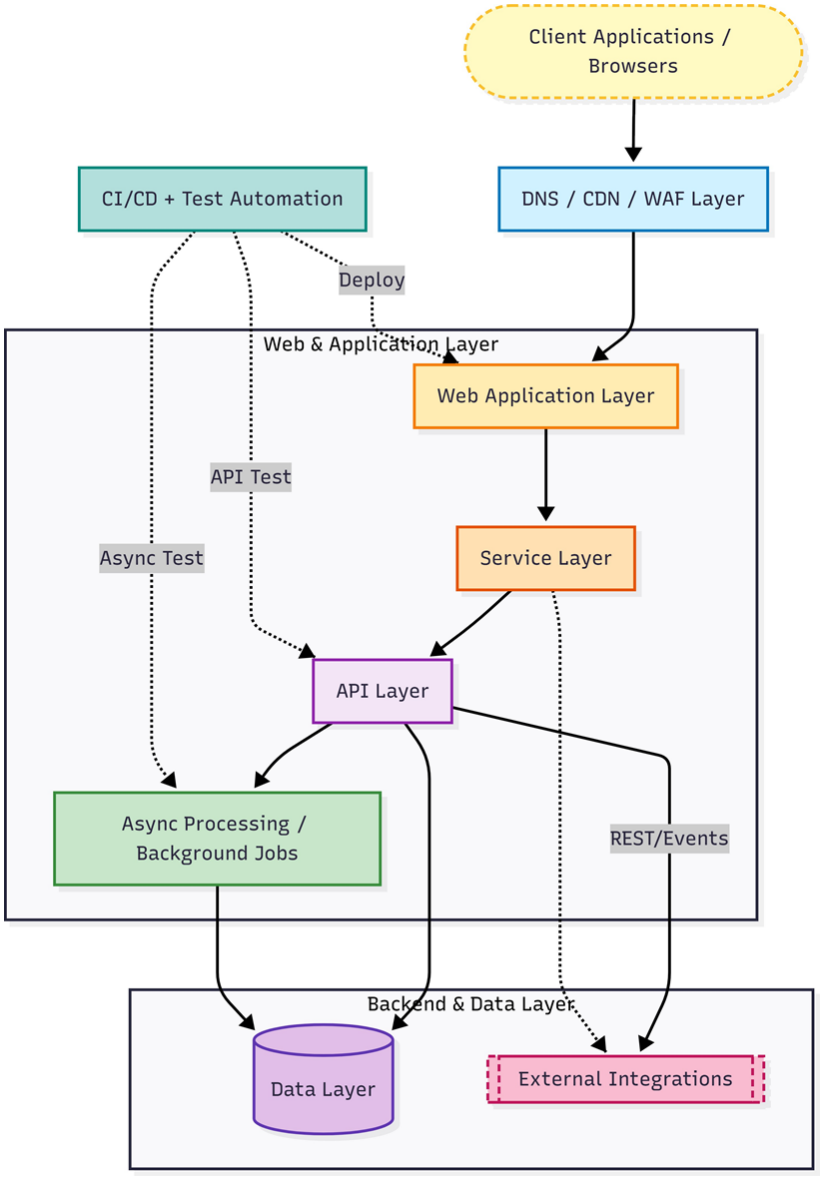

Figure 1: Generic Layered Web Application Architecture

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4 MANUAL TEST AUTOMATION FRAMEWORK (MTAF)

The Manual Test Automation Framework (MTAF) consists of six sequential phases designed to systematically transform large manual test repositories into sustainable automation pipelines.

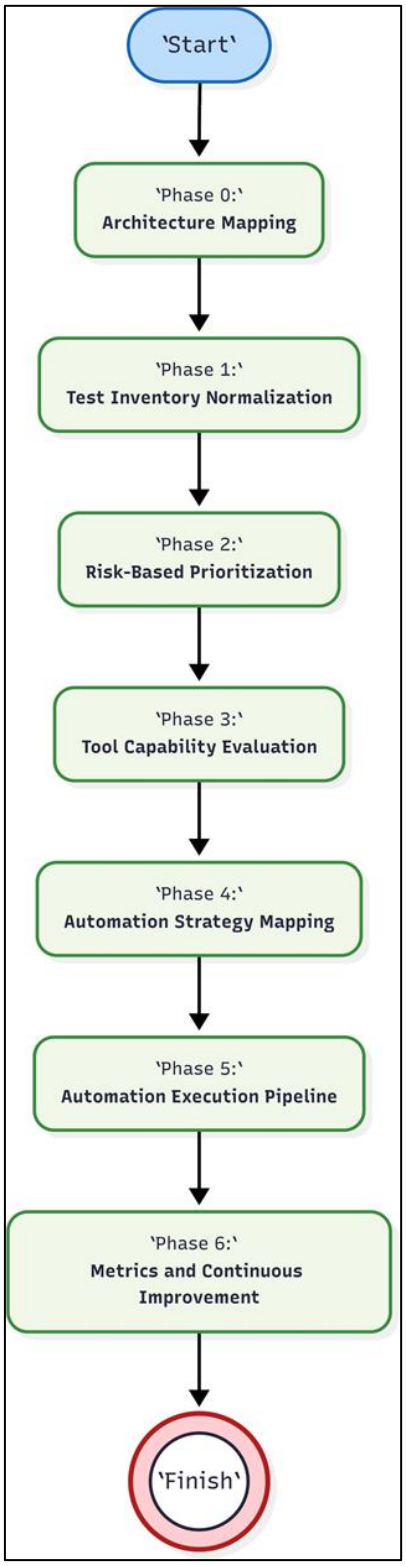

Figure 2: Manual Test Automation Framework (MTAF)

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4.1 Phase 0 – Architecture Mapping

The transformation began with explicit architecture mapping to establish a shared understanding of system structure, dependencies, integration points, and data flows. Manual test cases were mapped to architectural components, enabling traceability between regression coverage and system architecture.

4.2 Phase 1 – Test Inventory Normalization

Each test case was reviewed to assess relevance, duplication, and alignment with current system behavior. Obsolete and redundant test cases were removed, reducing the test inventory from 784 to 401 test cases.

4.3 Phase 2 – Risk-Based Prioritization

Remaining test cases were prioritized using a structured scoring model reflecting organizational risk considerations. The prioritization score (PS) for each test case was computed as:

$$PS = w_1 BI + w_2 DF + w_3 EF + w_4 WS + w_5 IC \quad (1)$$

where BI denotes business impact, DF historical defect frequency, EF execution frequency, WS workflow stability, and IC integration complexity. The weights w_i were determined through stakeholder alignment based on release criticality and risk tolerance.

4.4 Phase 3 – Tool Capability Evaluation

Initial automation attempts using traditional browser-driven frameworks proved difficult to scale. A structured tool capability evaluation was conducted based on cross-browser support, CI/CD integration, reporting maturity, extensibility, and ease of adoption.

4.5 Phase 4 – Automation Strategy Mapping

Rather than relying solely on UI automation, test cases were mapped to appropriate automation layers, combining UI, API, backend job validation, and integration testing.

4.6 Phase 5 – Automation Execution Pipeline

Automation execution was initially scheduled on a nightly basis. Integration with CI/CD pipelines and automated defect creation is ongoing. Automation-detected defects were labeled to enable early return-on-investment analysis.

4.7 Phase 6 – Metrics And Continuous Improvement

Key metrics included automation coverage growth, regression cycle duration, automation-detected defects, and cross-module defect discovery.

A defining characteristic of MTAF is its sequencing. By performing architectural mapping and repository normalization before automation execution, the framework reduces redundant automation effort and improves long-term sustainability through structural alignment of coverage and system components.

5 CASE STUDY RESULTS

The framework produced measurable improvements in regression efficiency and coverage.

Regression duration was measured as the calendar time required to complete a full validation cycle under comparable release scope conditions. Automated execution results were captured from nightly automation runs, and automation-detected failures were labeled to support early assessment of automation impact and return on investment.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Metric	Before	After
Total test cases	784	401
Automated tests	0	180
Regression duration	~12 days	3 days
Automation execution	None	Nightly

Table 1: Automation Transformation Results

Inventory normalization and prioritization were completed early to establish a stable roadmap, after which automation implementation proceeded incrementally over subsequent releases. This sequencing enabled early ROI by focusing initial automation effort on high-impact scenarios while continuing to expand coverage over time.

80% pass rate across nightly runs over a two-month observation period including failures attributable to environment instability and external integration dependencies. Given the distributed deployment model and mixed UI/API/backend test coverage, this level of stability was considered acceptable during the early automation maturity phase. Stability improvements remain an ongoing optimization objective. Most execution failures during this observation period were attributable to infrastructure instability or transient integration issues rather than deterministic test script defects.

Detailed defect density comparison and long-term maintenance cost analysis are outside the scope of this study and represent areas for future evaluation.

The initiative was executed by a small team consisting of one test engineering manager and three test engineers across multiple releases.

These quantitative results provide the empirical basis for the structural insights discussed in the following section.

6 DISCUSSION

The transformation initiative revealed structural, economic, and technical insights relevant to large-scale automation adoption in enterprise systems.

6.1 Structural Insight: Automation Feasibility Depends on Inventory Discipline

A key observation is that large manual test repositories often contain accumulated redundancy that obscures true regression coverage. Automation feasibility is constrained not only by tooling but by repository quality. The 47% reduction in test volume was not simply optimization but structural simplification. Without this reduction, automation scope would have remained fragmented and maintenance-intensive.

6.2 Economic Implications and Return on Investment

Test normalization required dedicated upfront effort. However, reduced regression duration and clearer automation prioritization generated downstream efficiency gains. The shortened validation cycle enabled faster release feedback and reduced manual overhead. The initial cleanup investment was offset within early release iterations due to measurable reductions in regression effort and maintenance complexity.

6.3 Technical Strategy: Layer-Appropriate Automation

Automation effectiveness improved when validation responsibilities were distributed across UI, API, and backend layers. UI-centric automation in layered architectures introduces fragility and maintenance overhead. Layer-appropriate placement reduced test instability and improved

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

execution efficiency. This observation suggests that automation strategy must align with architectural layering rather than default to end-to-end UI workflows.

6.4 Tooling as an Enabler, Not a Solution

The case study demonstrated that tooling decisions influence sustainability but do not compensate for structural deficiencies in test repositories. Structured evaluation criteria reduced future technical debt, but tool selection alone would not have delivered measurable improvement without prior prioritization and cleanup.

6.5 Governance and Sustainability

Test repository governance should be continuous rather than episodic. In evolving systems, review and pruning activities should occur at least once every two major release cycles in rapidly changing environments, or annually in slower-changing environments. Sustained inventory discipline prevents regression bloat and reduces long-term automation maintenance costs.

The framework has been formally proposed for reuse across additional application teams. Although broader implementation is still in planning phases, the initiative indicates perceived transferability beyond the initial case study.

6.6 Limitations

The findings are derived from a single industrial system. Although architectural characteristics are common in enterprise environments, empirical validation across multiple organizations would strengthen external generalization.This study does not quantify long-term automation maintenance cost or defect density reduction across multiple years. Future work should include comparative evaluation across multiple systems and longer observation periods to further validate maintainability outcomes and external generalization.

7 CONCLUSION

This paper presented the Manual Test Automation Framework (MTAF), a practical methodology for migrating large manual regression test suites into sustainable automation pipelines. The industrial case study demonstrated substantial reductions in regression execution time and expansion of defect detection coverage.

Future work includes deeper CI/CD integration, automated defect management, AI-assisted test generation, and predictive regression prioritization.

REFERENCES

[1] Garousi, V., Mäntylä, M.V.: When and what to automate in software testing? A multivocal literature review. Information and Software Technology 76, 92–117 (2016)

[2] Utting, M., Legeard, B.: Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann, Burlington (2010)

[3] Kochhar, P.S., Thung, F., Lo, D.: Code coverage and test suite prioritization in continuous integration environments. Empirical Software Engineering 24(3), 1773–1807 (2019)

[4] Tufano, M., Watson, C., Bavota, G., Poshyvanyk, D.: Using deep learning for automated software testing. IEEE Software 39(5), 46–53 (2022)

[5] Elbaum, S., Malishevsky, A.G., Rothermel, G.: Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering 28(2), 159–182 (2002)

[6] Fewster, M., Graham, D.: Software Test Automation: Effective Use of Test Execution

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Tools. Addison-Wesley, Harlow (1999)

[7] Rothermel, G., Harrold, M.J.: A safe, efficient regression test selection technique. ACM Transactions on Software Engineering and Methodology 6(2), 173–210 (1997)

[8] Yoo, S., Harman, M.: Regression testing minimization, selection and prioritization: A survey. ACM Computing Surveys 45(2), Article 14 (2012)

[9] Bavota, G., Qusef, A., Oliveto, R., De Lucia, A., Binkley, D.: An empirical analysis of the distribution of unit test smells and their impact on software maintenance. Journal of Systems and Software 85(11), 2501–2514 (2012)

[10] Leotta, M., Ricca, F., Stocco, A., Tonella, P.: Capture-replay vs. programmable web testing: An empirical assessment during test case evolution. Empirical Software Engineering 19(4), 916–947 (2014)

[11] Shahin, M., Babar, M.A., Zhu, L.: Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. IEEE Access 5, 3909–3943 (2017)

[12] Garousi, V., Felderer, M., Hacaloğlu, T.: Software test maturity assessment and improvement: A multivocal literature review. Information and Software Technology 85, 16–42 (2017)

[13] Myers, G.J., Sandler, C., Badgett, T.: The Art of Software Testing, 3rd edn. Wiley, Hoboken (2011)

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.