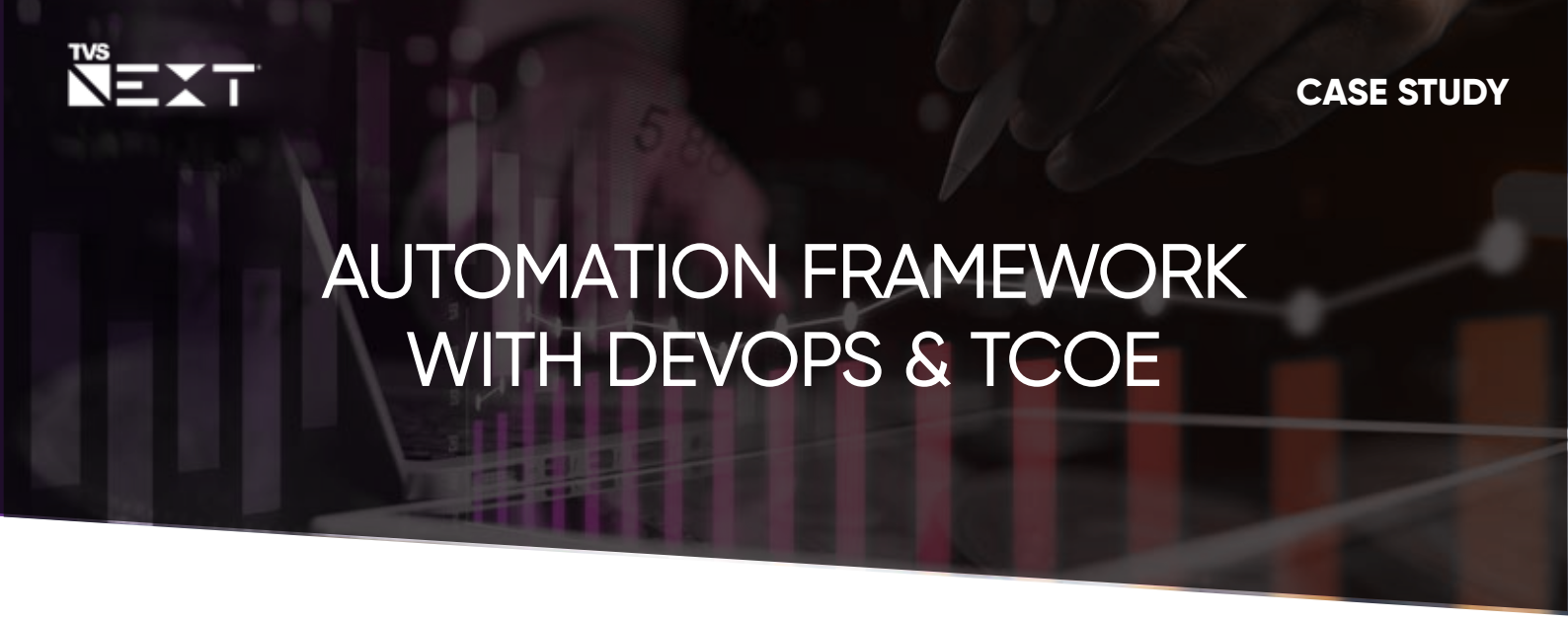


CASE STUDY

AUTOMATION FRAMEWORK WITH DEVOPS & TCOE

For a while the process of software-testing has adopted a shift-left approach, testing simultaneously with the development of CI/ CD. But test automation and shift-right approaches must be implemented to achieve quality at pace with agile and Devops on the go.

OVERVIEW

Our client focuses on loyalty as an outcome through its customer engagement SaaS platform. They empower world's most innovative brands to forge stronger and more profitable customer relationships.

The platform leverages data-driven loyalty execution, scales for the enterprise globally, and powers restaurants, retail, Consumer Packaged Goods (CPG), travel, and hospitality brands.

Businesses leverage this platform for defining their reward and loyalty management programs through various customer touch points like new purchases, incremental buys, absence from shopping et al.

The client's aspiration is to be the go-to brand to increase customer loyalty through delivering superior personalized experiences rooted in customer data.

CHALLENGE

In-spite of successful platform adoption, our client had tough time meeting deadlines for platform releases, customer specific changes and integrating continuous feedback. Major platform releases were slated half yearly. In addition to this every major customer deployment took another six months for implementation.

As the platform was adopted by customers in different geographic locations, extensive localization and internationalization was required, opening feature adoption challenges impacting user experience.

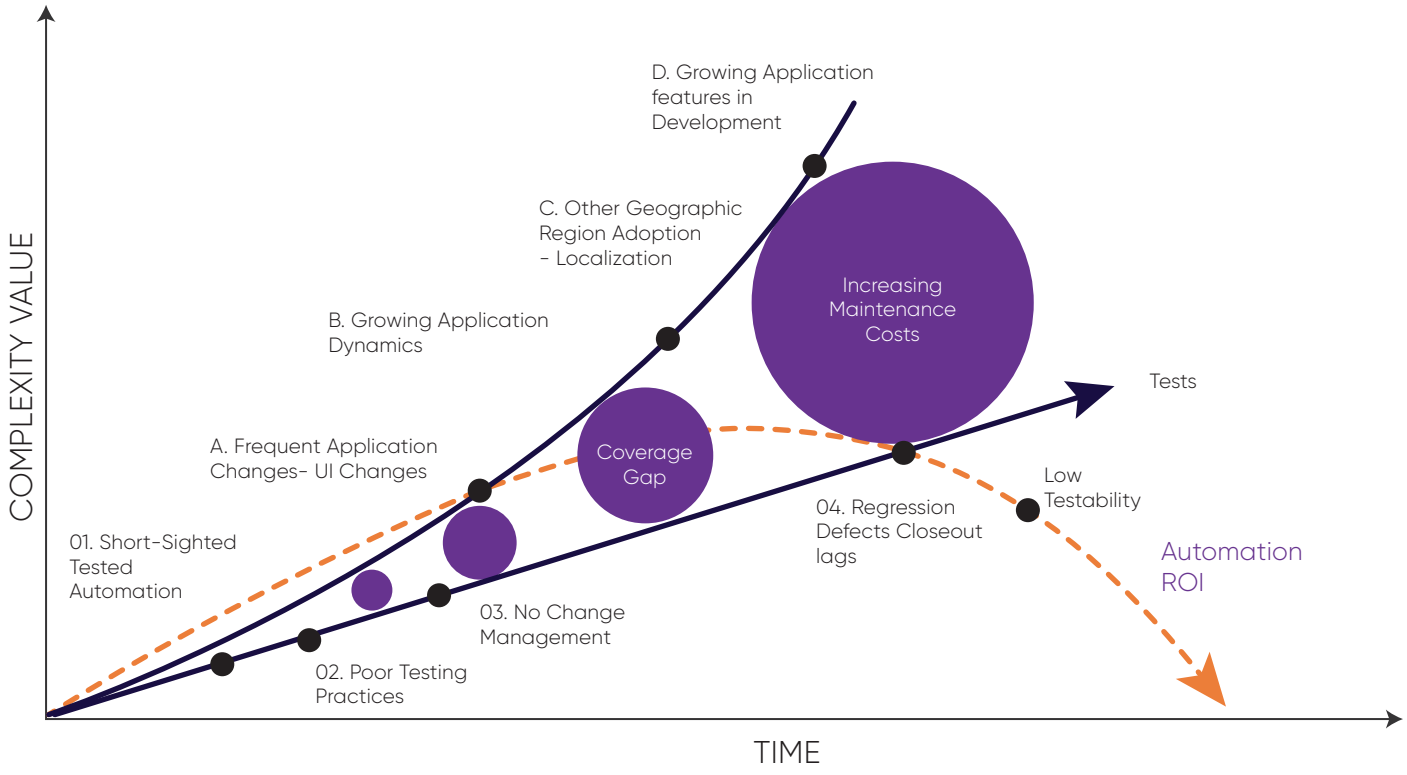
Therefore, the main challenge was long release cycles and locked-up time before feedback data was used to improve the platform.

ASSESSMENT

Through methodical assessment and maturity checks, the root cause of the challenges was identified. The assessment was structured to carry out development, testing and deployment stages. The technology stack and tools adopted were in alignment with processes, culture and people.

PRODUCT DEVELOPMENT, TESTING & RELEASE PHASE

The plot between the development stages across releases and the testing methodology is shown in the graph below.



While feature development and feedback was methodically, testing could not scale at this speed. The test case suites were not updated progressively, creating huge test coverage gaps.

API Integration of POS systems was not performed, leaving integration vulnerable.

One of the root causes of the long release cycle time was the stagnant test cases and lack of an agile testing practice. This required a structured and collaborative process from DevOps to QA and QA to delivery.

Existing test automation was dependent on UI elements. But ongoing releases were constantly changing the UI, making test automation framework not extensible and causing test engineers to rework.

TECHNOLOGY STACK & TOOLS PHASE

The core development used Java and 60% automation test cases where done with Selenium. Unit test cases were written in TestNG.

Tech stack across development and QA were from a common family. However, test cases required re-work as the testing framework was developed using ID based automation. With every UI feature change or implementation, the referring test cases required an update.

The tools and programming language did not require change but testing framework and methodology required an update. This leveraged the skillset of existing team.

CULTURAL AND PEOPLE PHASE

The organization had an open culture with an inclination to open source technology and learning. The culture was conducive for exploring other open source or related technology extensions.

New technologies introduced were preferred to be feature rich, adoptable, stable, and an open source solution. In case a licensed tool required enhancement, the culture encouraged new releases.

SOLUTION

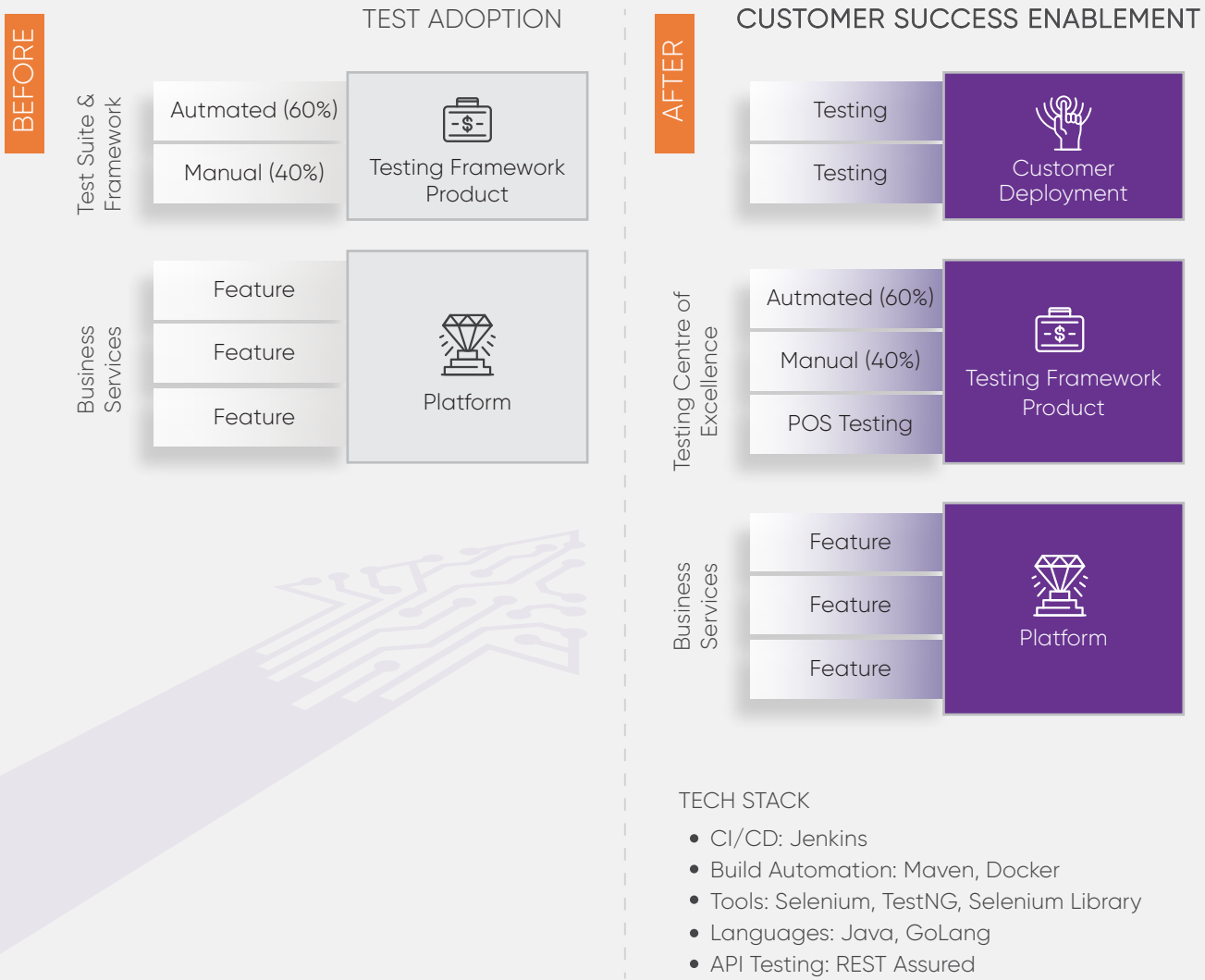
- We improved testability by replacing UI element dependent automation by Page Object Model. This reduced rework and enabled test case creation for every new feature.
- Delivered Speedy Release Cycles through Introducing CI/CD Pipeline for Automation Testing. The client had a very exhaustive and robust platform providing great value to its customers through their retainer, insights, and means. However, time to market was very steep for any enhancements or fixes. We methodically implemented the DevOps CI/CD process for the testing cycle, starting from creating a pipeline in Jenkins to automating their entire testing solution.

  As an outcome 2 weeks of manual testing effort reduced to sequential execution of the automated test cases in just 10 hours. To speed up the process, our team improvised it to a parallel execution approach which further reduced the turnaround time to 4 hours.

Through the assessment we identified the root cause to solve problem areas, which were low testability and development to QA collaboration.

- After establishing speedy release cycles, expanding business to different geographies became possible. To enhance customer experience, the platform required a UI upgrade and deep enhancements. This impacted the entire testing automation that was implemented based on the UI.

THE TRANSFORMATION JOURNEY



TECH STACK

- CI/CD: Jenkins
- Build Automation: Maven, Docker
- Tools: Selenium, TestNG, Selenium Library
- Languages: Java, GoLang
- API Testing: REST Assured

- As soon as the coverage gaps were bridged, we assisted in helping setup the “Testing Centre of Excellence”. We used the agile method to deliver the best results for our clients.

Our team pro-actively modified UI based testing on the Page Object Model framework thereby making every feature upgradable to the UI without affecting release cycles.

RESULT

We assisted in setting up Testing Centre of Excellence and pro-actively predict customer testing as an end-to-end TestOps partner. We helped them deploy customizations, test clearances, support, and bridge feature adoption to quality standard through automation.

- The maturity of the organization improved through partial **(35%)** DevOps adoption.
- The Parallel Execution Approach on modules helped reduce the test cycle from **96hrs** to just **4hrs**.
- We adopted Sequential Execution Approach to reduce the testing time from **96hrs** to **10hrs**.
- This approach also helped decrease **95%** test cycle time and increase speed of release time.

Shorter release cycles and intermittent releases were introduced. The minor releases happened every month to fix bugs and the major releases happened every three months for new feature addition and system enhancement.