

**View Point**

## **Pillars of Quality Engineering**



## The need for quality engineering services

In today's digital world, 'change' is the sole constant and organizations are grappling with ways to fulfill the dynamic expectations of customers. When organizations embark on digital transformation in addition to changing market dynamics — Creating remarkable customer experience built/incorporating new & emerging technologies such as AI, RPA, IoT, API integration, smart payment solutions, etc., and that which is unified across a growing range of applications, devices, platforms, and interfaces — is turning into a herculean task. Moreover, emphasis on risk and compliance is also reshaping expectations from IT. To attain the specified pace and adaptability that enable shorter release cycles and quicker time-to-market, organizations are currently adopting agile strategies and DevOps principles.

This necessitates the need to remodel from a typical Quality Assurance (QA)-driven function to one that's Quality Engineering (QE)-focused.

QE not only embeds quality in development of product and processes but also enables effective testing in parallel. The idea is that during the software development life cycle, the test automation strategy is designed beforehand and then the developers, operations and QE teams, all work together as a single unit. Automate the test cases beforehand, script them, and make the executables ready as part of the development process.

### Distinction between QA and QE

Quality Assurance "assures" quality of the product, but quality engineering drives development of quality product and the processes. This includes not only quality, maturity of the quality team itself, but also indicates a cultural shift within the teams. Also, it includes agile-based processes and uses Test-Driven Development (TDD) and Behavior-Driven Development (BDD) processes to define requirements that are meaningful to the business developers and the testing teams.

Quality Engineering (QE) is the process by which software quality is tested, analyzed and improved throughout the application development lifecycle. Quality engineers focus on the quality right from the requirements stage, and use tools like FMEA (Failure Mode Effect Analysis) to brainstorm what might cause failure and then implements a quality system to prevent those failures from occurring or allow detection of the failure. The push is to start integrating quality across the entire SDLC to engineer quality. The test framework requires a common set of tools to be used, so that any test data manager can assist on running the tests across multiple devices simultaneously. These automated tests are being culled from continuous integration and continuous deployment processes, and being validated as part of build deployment.


### Pillars of QE transformation

#### QE across the lifecycle

Quality Engineering across the IT lifecycle helps deliver the committed business benefits across various technology initiatives.

Re-engineering QA function into an integrated unit with development enables to deliver sustainable quality and improved velocity, in line with business requirements by shrinking the post development test phase and release cycles.

The software paradigm is called shift-left wherein the emphasis is to move software quality activities to the beginning of the software development life cycle during conceptualization and development.



A crucial norm for “Shift Left” is to write testable code that is unit verified, and build quality at a component level up to integration level. Essentially this enables efficient localization of a problem and ensures that individual elements are working before a large amount of software is integrated.

With QE, the emphasis is on creating a lot of automated test cases closer to the development code. So, a large set of unit test cases closer to development code are developed, and there’s lots of area for automated test cases and the competent technician at the API test level as well as lots of UA-related test cases. This enables rapid development or verification very easily in a heightened manner, in order to complete validations very quickly. When compared to legacy software development lifecycles, there is more manual testing done and less emphasis and data on unit tests. The result is additional time and cost to verify across multiple devices.

In the new model, a lot less time is spent on manual testing, and there are more automated test cases – closer to the development code. This should provide tighter integration of automated test cases closer to development code.

### CI/CD Infrastructure

Moving to an effective and efficient CI/CD pipeline requires significant effort from organizations. It involves process and policy changes across the entire organization. The payoff can be extraordinary: continual improvement as the organization moves to constantly deliver high-quality, well-tested value to their customers.


Adding integration and end-to-end testing to a pipeline can enable the leap from Continuous Integration to Continuous Delivery (CD). Being effective, long term, with Continuous Delivery requires an entirely new set of features -- testability features -- built into the architecture itself, along with other changes to the way software is built. Without these changes, organizations typically struggle to see the benefits that Continuous Deployment and Delivery promise. Organizations looking to move to Continuous Delivery would do well to consider improvements in the following categories, looking for “missing features”, anticipating the cost of building those features, and the consequences of leaving them blank.

- Efficient Management of Environments
- Management of features
- Adapting Processes
- Implementing Codebase Improvements

### Integrated Coverage

- Comprehensive management of four key components across the IT lifecycle
- Application stack (Single, dual or three tier architecture),
- Interfaces,
- Data, and
- Configuration.

Once process and policies have been configured to adopt this movement to CI/CD-driven Quality Engineering, integrated coverage is essential in order to improve efficiency of the entire QE process through comprehensive management of all the enablers in the SDLC. Traditional QA techniques are methodologies are enhanced and utilized to improve efficiency of a QE-enabled lifecycle.



For example, full-stack quality engineering teams are being groomed by organizations to ensure quality across all layers of the application using assurance techniques; interface management typically involves effective communication across now-integrated teams on all aspects concerning an interface including requirements, technical specifics, as well as potential problems; data management is often seen practiced in the form of utilization of data to prevent problems and further improve quality, which also leads to the need for sound configuration management to maintain system integrity in the light of changes wrought forth through implementation of application stacks, large data-sets, and multiple interfaces.

### Dashboard for predictive analytics

Metrics are a way to measure and monitor test activities. More importantly, they give insights into the team's test progress, productivity, and the quality of the system under test. The metrics need to cover various aspects such as the source code used, the process that is put in place for the custom development, test coverage and the defect removal efficiency. The dashboard for predictive analysis should cover the following.

- Collection of metrics on requirements, development and QA effectiveness
- Use of internal data relevant to development and QA such as system, operational logs and defects
- Use of external data where relevant
- Predictive models for Impact, Risk and Defects
- Dashboard across the lifecycle on all the above.

### Changing Mindsets from Quality Assurance to Quality Engineering

If one's organization or software is going through the digital transformation journey, then the terms Agile, DevOps, shift-left, etc. will become more prominent. There is an important transition of mindset from Quality Assurance to Quality Engineering to manage the overall of quality effort for these rapidly scaling applications. It means that there is a brand new approach to software testing in the making. Clearly, because the old methods were not able to support rapid, zero-defect feature releases on time, and for banking applications which are created to serve thousands of users.

Some of the frameworks have made development and quality teams dysfunctional. There are many rejects to QA that have caused a lot of development rework. Result – very few software releases, huge overhead, and loss in market shares. So there's an immense rethinking and restructuring of all the frameworks, reports, tools, and iterative frameworks.

## ABOUT MAVERIC

Started in 2000, Maveric Systems helps global banking and fintech leaders drive business agility through effective integration of development, operations and quality engineering initiatives. Our strong banking domain competency combined with expertise across legacy and new age technology landscapes makes us a preferred partner for customers worldwide.

We offer Product Implementation, Integration and Quality Engineering services across Digital platforms, Banking solutions and Regulatory systems. Our insight led engagement approach helps our clients quickly adapt to dynamic technology and competitive landscapes with a sharp focus on quality.

**Maveric Systems Limited (Corporate Office):** "Lords Tower" Block 1, 2<sup>nd</sup> Floor, Plot No 1&2 NP, Jawaharlal Nehru Road, Thiru Vi Ka Industrial Estate, Ekkatuthangal, Chennai 600032

India | Singapore | Saudi Arabia | UAE | UK | USA | Mexico

Write to us at [info@maveric-systems.com](mailto:info@maveric-systems.com) | [www.maveric-systems.com](http://www.maveric-systems.com)