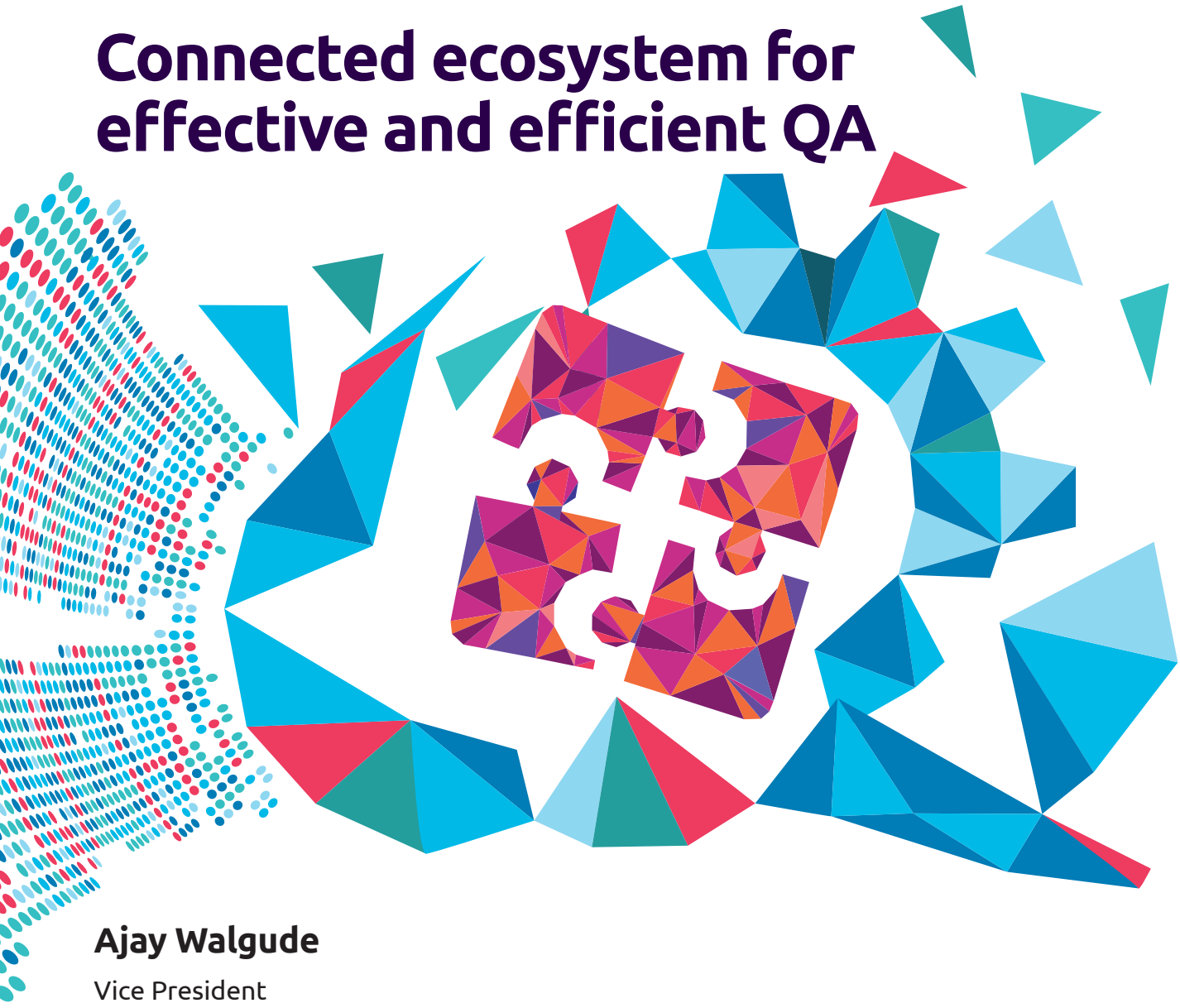
Key trends in Quality Assurance

Connected ecosystem for effective and efficient QA



Ajay Walgude

Vice President
Capgemini Financial Services

To adopt to new ways of working – such as Agile, bi-modal or uni-modal, many organisations have moved away from a horizontal, centralised testing function to a vertical model, aligned to business units, tribes or transaction cycles.

This new world order is either fully decentralised and aligned to verticals or to a hybrid, federated structure, where part of the QA/QE organisation acts as a quality centre of excellence (QCoE) responsible for quality control, governance, best practices and tooling management. As well as a QA Guild for specialised testing such as performance engineering, automation framework and security testing, there are pros and cons for each of these approaches.

There has been a change in the way testing is organised and performed, the skills required for testing, the tools used, and what is expected from QA as a function. This impacts on when testing gets engaged and the techniques being used to carry it out. The biggest difference has been the behavioral change upstream to enable that, the change in buying pattern for the services from service providers as well as the skills and tools change.

Everything seems to be in place or getting in order, but we still have lower defect removal efficiency (DRE), high cost of quality especially the cost on non-conformance based on interactions with various customers. While the World Quality Report (WQR) acknowledges these changes and comments on the budgets for QA being stable or reduced, there is no credible source that can comment on metrics such as cost of quality, and DRE across phases, and type of defects (the percentage of coding defects versus environment defects).

The above alignment of QA teams was associated with achieving quicker time-to-market by means of frequent and incremental feature release cycles, building quality in the product by shifting left and eliminating defect at source. To make sure it works and deliver the results as designed, it needs a fully connected ecosystem for QA. Dev and QA assets – such as requirements, code, build and deploy kits, test cases, automation scripts for continuous testing, for purpose test environment and test data provisioning – should be seamlessly integrated in the ecosystem.

These ecosystem elements are leveraged and managed by multiple parties including product owners, BA, dev, test, and infrastructure release management teams. Everybody plays their part in ensuring they are available when needed but they may not be always in sync or fit for purpose. For example, some elements in the environments may be wrongly configured, integration pieces may be missing, or the environment may not have all the data required – especially for new features being introduced. The QA team may also be unaware of what has changed, the code itself, the business logic or the configuration parameters.

A successful ecosystem isn't just about communication where the teams exchange information with other stakeholders on what is being changed or required to effectively test. The key factors in building a successful connected eco system include:

- **Every team member must be willing to learn the entire system**. This includes architectural details, use cases, code base, configuration requirements, dependencies, and the business domain. Team members do not need to be experts in all of these areas, but they should have a basic knowledge that allows them to communicate with stakeholders.
- **Updated documentation and mapping of various entities.** It may sound like a lot of unnecessary work because of the segregation of duties where everybody focuses on their part of their work and are co-located mostly in a pod structure, but this establishes the base before the smart assets are built.
- **Building smart assets** that will understand the connection between the entities described above and provide actionable insights. For instance, a change in the application configurations parameters might trigger a note to the environment engineer indicating changes that are needed in the environments. Another example would be specific changes in a code base automatically selecting the right test cases to be executed by focusing on targeted and affected areas. Simply put, this is an automated workflow of release notes that triggers a sequence of action items based on the change and notifies the corresponding stakeholders.
  So far, we have discussed the impact of the left of the software development life cycle (SDLC) but even the right part – such as production behavior and patterns – can influence and determine the testing strategies or data and environmental decisions and code changes.
- Build bots that can take action unattended and in a predictive manner. If we build what I call Actionable Architecture which maps all of its entities even a small piece code to larger programs and its dependencies. If this is implemented right, it is possible to build code scraper bots and feedback bots to understand or predict changes, relational bots to determine the impact and next best action because of the change, and action bots to execute these programs.

Effective and efficient QA relates to testing the right things in the right way. Effectiveness can be determined in terms of defect and coverage metrics such as defect removal efficiency, defect arrival rate, code coverage, test coverage and efficiency that can be measured in terms of the percentage automated (designed and executed), cost of quality and testing cycle timeframe. The connected eco system not only has a bearing on the QA metrics – cost of appraisal and prevention can go down significantly – but also on the cost of failure.

The aim of squads, tribes, pods and scrum teams is to bring everybody together and drive towards the common goal of building the minimum viable product (MVP) that is release worthy. While the focus is on building the product, sufficient time should be spent on building this connected eco system that will significantly reduce the time and effort needed to achieve that goal and, in doing so, addressing the effective and efficient QA.

Conclusion: While businesses automate testing activities and apply artificial intelligence other solutions for efficiency and effectiveness of testing, a connected eco system made up of smart assets, that is self-aware and can steer itself to be fit for purpose, is required to realise the potential of these initiatives.