Quality at the speed of digital

Modern testing using a cognitive approach

Introduction

Quality has always been at the backburner despite being the most critical part of the software development process where delivering a high performing application is priority. With businesses coping with a dynamic and demanding marketplace, the pressure to deliver high quality software is never ending. Agile methodology has taken the industry with a storm, especially the IT industry. Apart from customer satisfaction, reduced risks, and quick ROI, agile helps in:

1. Stakeholder engagement
2. Transparency
3. Early and predictable delivery
4. Predictable costs and schedule
5. Flexibility
6. Quality

The World Quality Report from CapGemini, Sogeti and Hewlett Packard Enterprise provides a detailed view of just how Quality Assurance (QA) is changing. The process of digital transformation and shifts to agile and constant DevOps management models are changing how QA and testing is carried out. Moreover, the demands placed on the test function in enterprises are widening and changing too. Testing departments looking at these areas of business are also expected to provide intelligence for business development and growth, not just report on past performance.

This paper will look at what the traditional QA cycle offers and its challenges, how we can streamline QA process with the current development life cycles (Agile) using a cognitive testing approach and the 'LEFT SHIFT'.

Table of Contents

The traditional approach and its challenges

The traditional approach of QA in the software development process will involve the following:

1. Ensuring application functionality
2. Functional application automation
3. Workflow: Development, Testing, Ops
4. Descriptive metrices used for monitoring and control

Challenges in agile software development pertaining to Quality Assurance and Validation, such as time constraints in development and test cycles, executing large volumes of test cases and testing diverse legacy applications remains unanswered even after coming a long way since the industry has been introduced to agile methodology.

A constant need of a business is to reduce the time between development and delivery. A QA expert must evolve and transform which requires a paradigm shift from a conventional QA to what we call 'QA in agile world'. The challenges these days go beyond the technology stack. The following are the pain areas in a traditional QA process:

Technical complexity

The technology and platform stack is not limited to traditional desktop and the web for current application portfolios. It extends to multiple OS (platforms), mobile devices and the most cutting-edge responsive web applications. The challenge lies in complete testing of an application on all platforms within the give deployment cycle.

Infrastructure, licensing and training costs

To test diverse applications, multiple test automation tools need to be procured (license cost), testing environment needs to be setup (infrastructure setup cost and time), and the technical skills of the team need to be up to speed through trainings and self-learning/ experimentation (incurring costs and time).

Inadequate test coverage

With continuous integration and changing requirements, critical tests for any requirement are easy to miss. Another reason could be changing the code that was not anticipated. Due to less development time in the agile environment, developers often refrain from writing Unit Tests which means poor code coverage, resulting in deficient code coverage and the end result being poor quality software delivered to the customer.

Performance bottlenecks

As software becomes more mature, complexity increases. This complexity adds more lines of code which introduces performance issues if the developer is not focused on how their changes are impacting end-user performance. It becomes important for the QA team to focus on each module to make sure application can take proper load once live.

Late involvement of users

The end user is not involved in the development process until acceptance testing and is totally unaware of whether the implemented systems meet user requirements. There is no direct traceability between requirements and implemented system features. User validation goes for a toss if there is late involvement of the user resulting in re-work which involves additional costs as well as time, moreover delays time to market.

Late involvement of testers

Often QA is involved in the later part of the development cycle which often delays deliverability. For example, API testing and BDD approach needs to start earlier in the life cycle with agile achieved through the amalgamation oftechnical and domain skills of the team, as well as the end user.

As we can see traditional QA process are no longer sufficient in overcoming challenges that we face while developing in new and agile software development lifecycles. We need a modern approach to ensure seamless testing and quality assurance in the agile development lifecycle.

Quality Assurance in an agile world: Digital testing using cognitive approach

To meet the current business requirements, we have reformed the QA strategy. We need to move away from the traditional QA approach, and embrace 'QA in agile world' and DevOps. The various dimensions which need to be taken care of going ahead include:

1. Business value
2. Customer-centricity
3. Industrialization of security testing
4. Predictive analysis to prioritize testing activities
5. Automating the automation
6. Virtualization and cloud testing platform capabilities
7. Expand the QA skill set. QA -> QE -> SET -> SDET

Our approach to speed up the current QA practices to digital testing using cognitive approach includes:

- Machine Learning
- Artificial Intelligence
- DevOps
- Automation

The modern approach answers or helps in:

- Getting the estimation to test new features
- Understand the associated risk
- Predict solution
- Value of cognitive testing

Plan	Define	Develop	Test	Release	Operate
Investment decision analytics	IDX-based requirements	Code quality analytics	Test optimization & Defect prediction	A/B Testing & Canary	Log & DX Analytics
Portfolio backlog prioritization & value analysis	Validated requirements in backlog	Predict failures based on code commits	Predict defects/ failures	Scenario identification based on trends	DX Scores
Improved scheduling & capacity	Improved estimates	Dynamic code promotion criteria	Optimized tests		Anomaly detection

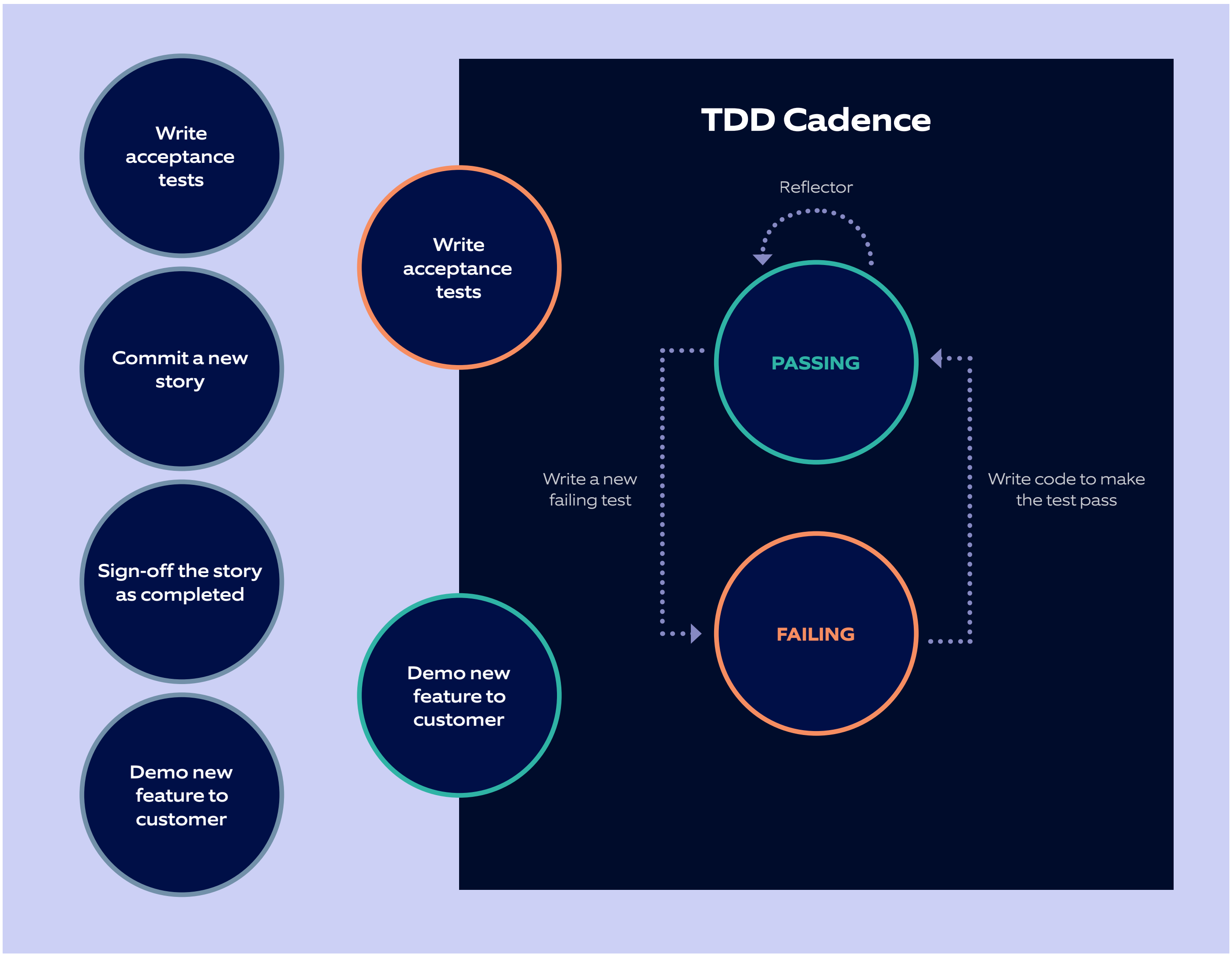
Key components of our digital QA approach

BDD approach and API testing: Automate the Automation

Behavior-Driven Development and Behavior-Driven Testing (BDD and BDT) is a way to reduce the gap between the end user and the actual software being built. It is also called 'specification by example'. It uses natural language to describe the 'desired behavior' of the system in a common notation that can be understood by domain experts, developers, testers and the client alike, improving communication. It is a refinement of practices such as Test-Driven Development (TDD) and Acceptance Test-Driven Development (ATDD).

The idea behind this approach is to describe the behavior of the system that is being built and tested. The main advantage is that the tests verifying the behavior reflect the actual business requirements/user stories and generate live documentation of the requirements, i.e. the successful stories and features as test results.

Therefore, the test results generated can be read and understood by a non-technical person for example a project sponsor, a domain expert, or a business analyst, and the tests can be validated.



API Testing: API is the brain of our connected world. It is the set of tools, protocols, standards and code that glues our digital world together. APIs enable companies to become more agile, for things to go mobile, and everything to work together in a streamlined and integrated manner. Moreover, by using API automation we can effectively use the API execution result to validate the functionality in one go after the API is developed.

While developers tend to test only the basic functionality, testers are in charge of testing functionality, performance and security of APIs, discovering how all components work together end-to-end. API testing should be done with all sets of data, testing techniques including Equivalence Partitioning, Boundary Value Analysis and Positive/Negative Data.

The API gateway is the core piece of infrastructure that enforces API security. Unlike traditional firewalls, API security requires analyzing messages, tokens, and parameters, all in an intelligent way. It checks authorization, then checks parameters and the content sent by authorized users as well.

API testing should test the performance of the application, like how much time is taken to login.

Benefits of API Testing

- Early detection of bugs
- Technology independent
- Protection from malicious code and breakage
- Test reuse
- Integration of API testing with CI/CD

Infrastructure Testing

The concept of infrastructure testing comes from the agile methodology where projects continues to make significant changes in revamping their infrastructure to keep pace with the performance of applications with change in frequent requirement. However, the lack of a structured approach to validate the underlying infrastructure leads to technical glitches and outages, resulting in severe business impact.

Infrastructure testing is an integrated solution that enables organizations to test and validate infrastructure components, thereby reducing the chances of downtime and improving the performance of their infrastructure.

Infrastructure testing is any test activity performed as a result of the setup of or an intervention made on hardware, network and/or software components which are part of an integral infrastructure platform to facilitate applications in a managed and controlled way. Infrastructure testing involve testing for the client, server, network, storage, and middleware.

Challenges and Opportunities

Across an organization, diverse technologies are used, including virtualization and cloud. The increasing number of technologies usage needs to be validated and streamlined to transform within the data centers. End-to-end infrastructure testing services with related program governance, risk assessment and performance will help organizations scale up to meet the challenge resulting from the multidimensional complexity of their growing infrastructure and application portfolios.

Benefits of Infrastructure Testing

- Unwarranted system failures, performance issues, and unplanned downtime
- Post deployment errors that lead to and require high cost of corrections
- Lack of standardized processes which gives out inconsistent results and deliverables
- Cyber-security is typically enabled at the infrastructure level
- Disaster recovery and Business Continuity Planning is critically dependent upon tested infrastructure
- Faster time to market with reduced testing cycle time
- Reduced cost due to accelerated testing using pre-defined testing libraries

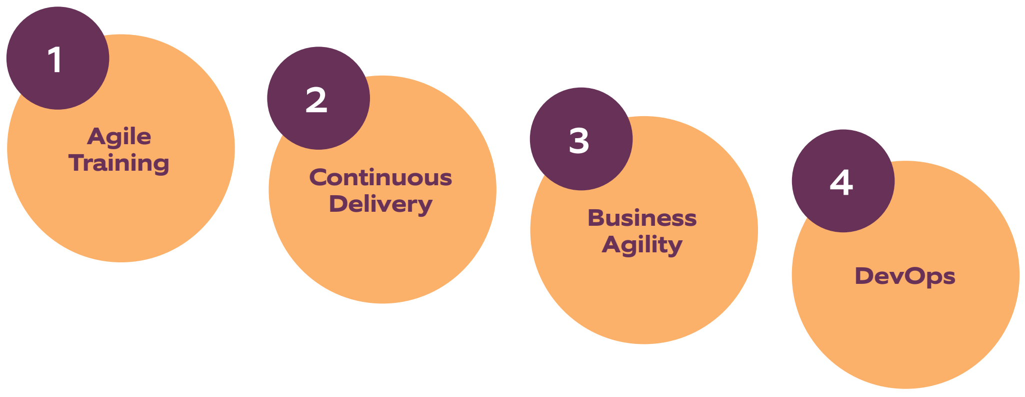
DevOps QA

Continuous testing acts as the key driver for DevOps initiatives to yield desired outcomes. This calls for specialists who understand the nuances of continuous testing through effective end-to-end automation leading to quality at speed. DevOps provides solutions that enable automated test orchestration and ensures rapid product development and deployment through an integrated model of continuous integration, continuous testing, and continuous delivery. Incorporating DevOps QA testing helps organizations develop a seamless development and production environment, armed with a continuous feedback made possible through continuous testing. Each stakeholder in all the areas pertaining to development plays a crucial role in our approach towards DevOps QA, whether they are involved in the stages of production, build, or release. This has the following implications.

- QA owns continuous improvement and quality tracking across the entire development cycle. They are the ones who are primarily responsible for identifying problems not just in the product but also in the process and recommending changes wherever they can.
- Tests are code, as any test automation expert will tell you. It's a necessity, of course. If your process is designed to publish a new release every day (or every hour) there is no room for manual testing. You must develop automation systems, through code, that can ensure quality standards are maintained.
- Automation rules. Anything that can be automated, should be automated. When Carl describes Unbounce's deployment process as "push-button easy," this is what he's talking about.
- Testers are the quality advocates, influencing both development and operational processes. They don't just find bugs. They look for an opportunity to improve repeatability and predictability.

DevOps is agile taken through its logical evolution, removing all the obstacles to getting high-quality software in the hands of customers. Once you have a smooth process for agile development and continuous integration, automating the deployment process makes total sense because it's achieving the objectives that business managers aim for:

- Faster time to market
- Better quality
- Increased organizational effectivenes

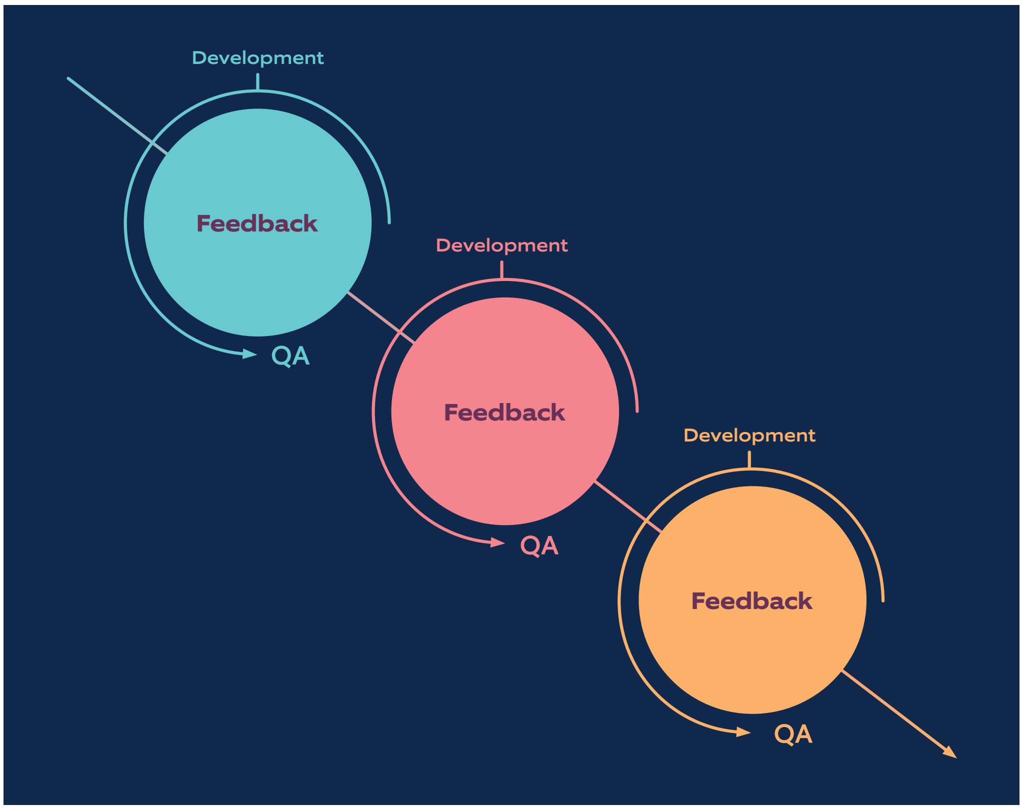


Parallel Testing

Parallel testing is the process of running multiple test cases on multiple combinations of operating systems and browsers at the same time. The process is automated, and it often runs on virtual machines.

QA and Development: Working in Parallel

In agile software development, where small teams are working fast and releasing new features frequently, this isn't the best approach. Being agile and delivering a good quality product in less time, QA and developers should be in sync, working together in parallel.

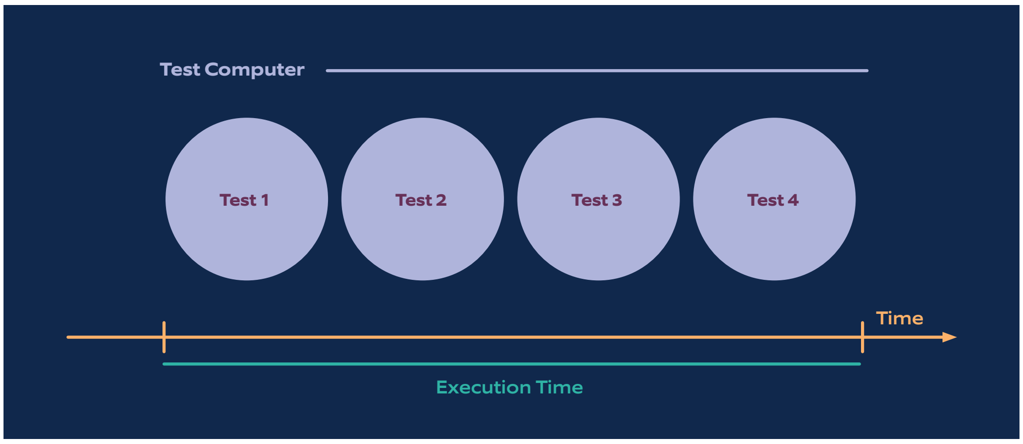


Parallel Testing with Multiple Devices

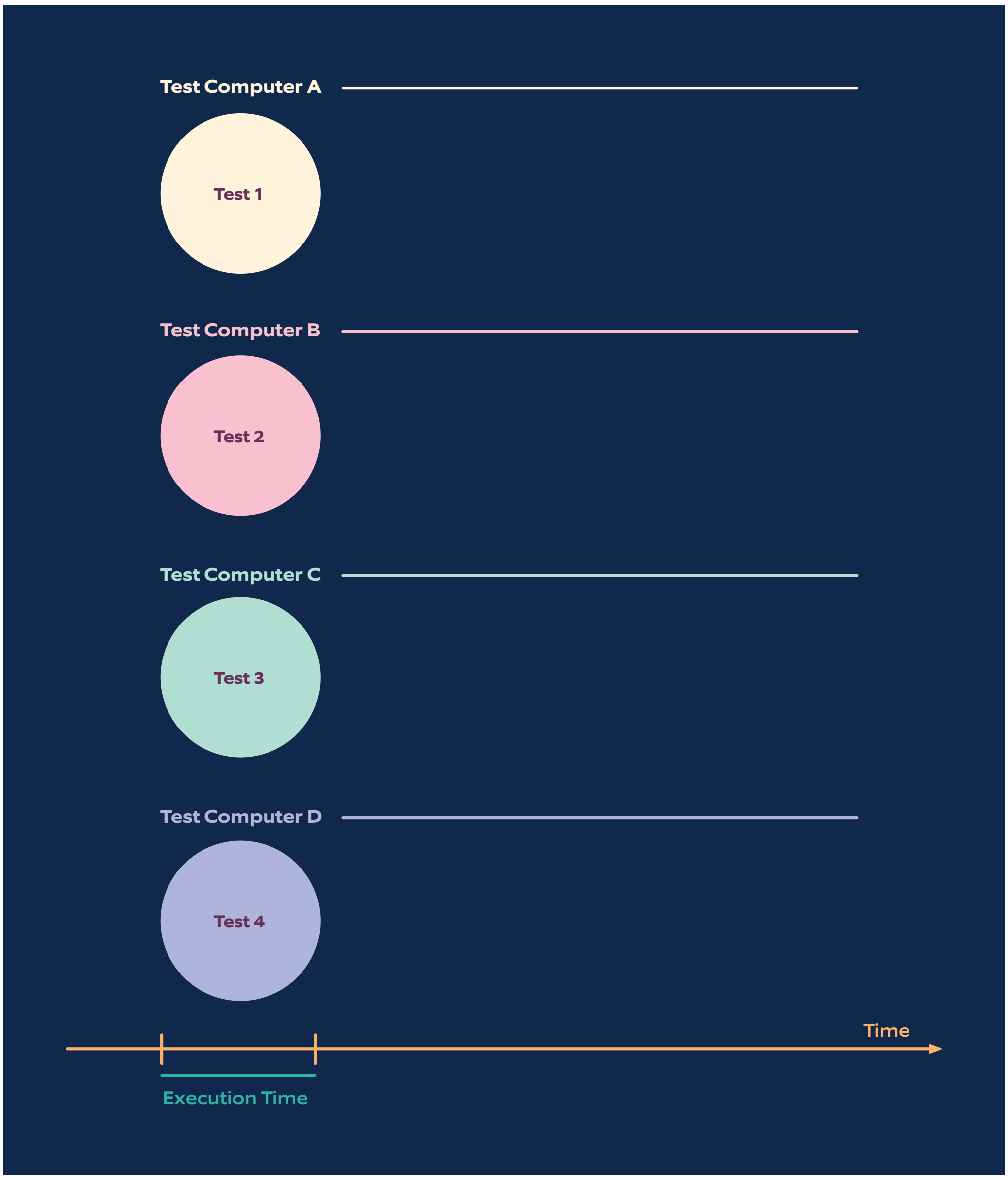
Using Power mapper tool, we can reduce the testing efforts from 80% to 20%. As Power Mapper or crossbrowsertesting.com tool allows you to test on different configurations, environments and operating systems. Product features which can be tested easily and covers below aspects of testing automatically:

- Broken links, server configuration errors and spell checking (English and French)
- Accessibility (WCAG 1.0 AAA, WCAG 2.0 AAA, and Section 508)
- Browser compatibility (Internet Explorer, Firefox, Safari, Chrome, Opera, iPhone/iPad)
- Privacy checking (including EU Privacy Regulations)
- Search optimization (Google, Bing and Yahoo Webmaster Guidelines, Robots.txt, Search Best Practices)
- Web standards (HTML, XHTML and CSS validation)
- Usability (Usability.gov Guidelines, W3C Best Practices)

Sequential Testing



Parallel Testing



Pairwise Testing/Combinatorial Testing

You cannot test everything with so many possible tests to choose from all, so we need to select just a few tests and still adequate the coverage. Greater coverage with fewer test cases is called Pairwise Testing or combinatorial testing or all-pair testing. To cover all the combination of parameters interacting with each other need to be tested and to identify smartly we use pairwise testing. Pairwise testing is most effective way of optimizing number of test cases without impacting on coverage. It executes a pairwise test data set.

How to create pairwise test sets:

a. Generating using Orthogonal Array

b. Algorithmic generation

Number of Inputs	Number of selected test data values	Number of combinations	Size of test size
7	2	128	8

How to create pairwise test sets:

a. Hexawise

b. AllPairs

c. PICT (Pairwise Independent Combinatorial Testing Tool)

d. Automatic Efficient Test Generator

Test Case Optimizer

Based on the concept of orthogonal array, our Test Case Optimizer is an innovative tool designed to improve productivity and reduce the test lifecycle significantly. It can help:

- Boost productivity by reducing test cycle time
- Increase test coverage
- Optimize the size of a test pack by rejecting redundant test cases, reducing test design and execution effort

Limitation of pairwise testing

- Selecting wrong inputs
- High probability combination gets little attention
- Not knowing the interdependencies between variables

Predictive Analysis of Metrics Value for Monitoring and Control

Predictive analysis is a practice of analyzing useful information from data to determine the trend and predict the outcome pattern using algorithm and machine learning.

Some of the key analytics approach and use case

1. Descriptive Analytics
2. Diagnostic Analytics
3. Predictive Analytics
4. Prescriptive Analytics
5. Deep Learning

Descriptive & Diagnostic Analysis would answer the root cause for past defects, predictive analytics would help in what defect we like to find with some Prescriptive Analysis and Deep Learning for undiscovered errors and what preventive measure should we take.

Key reasons for Predictive Analysis for Software Testing

- Build customer-centric QA using customer feedback analytics. Listen to social feedback to formulate QA process.
- Facilitates insights for prioritizing testing activities
- Predict right environment/right tools and boost testing efficiency
- Reduces overall cost due to early defect detection

Independent system testing takes about 20 to 50 % of the development time depending on various combination of types of testing conducted on a product. The complex the testing gets, more would be the effort required. The more a product is tested, the more quality it would be. Typically, test result reporting would consist of about 5% to 10 % of this effort.

Comparative view

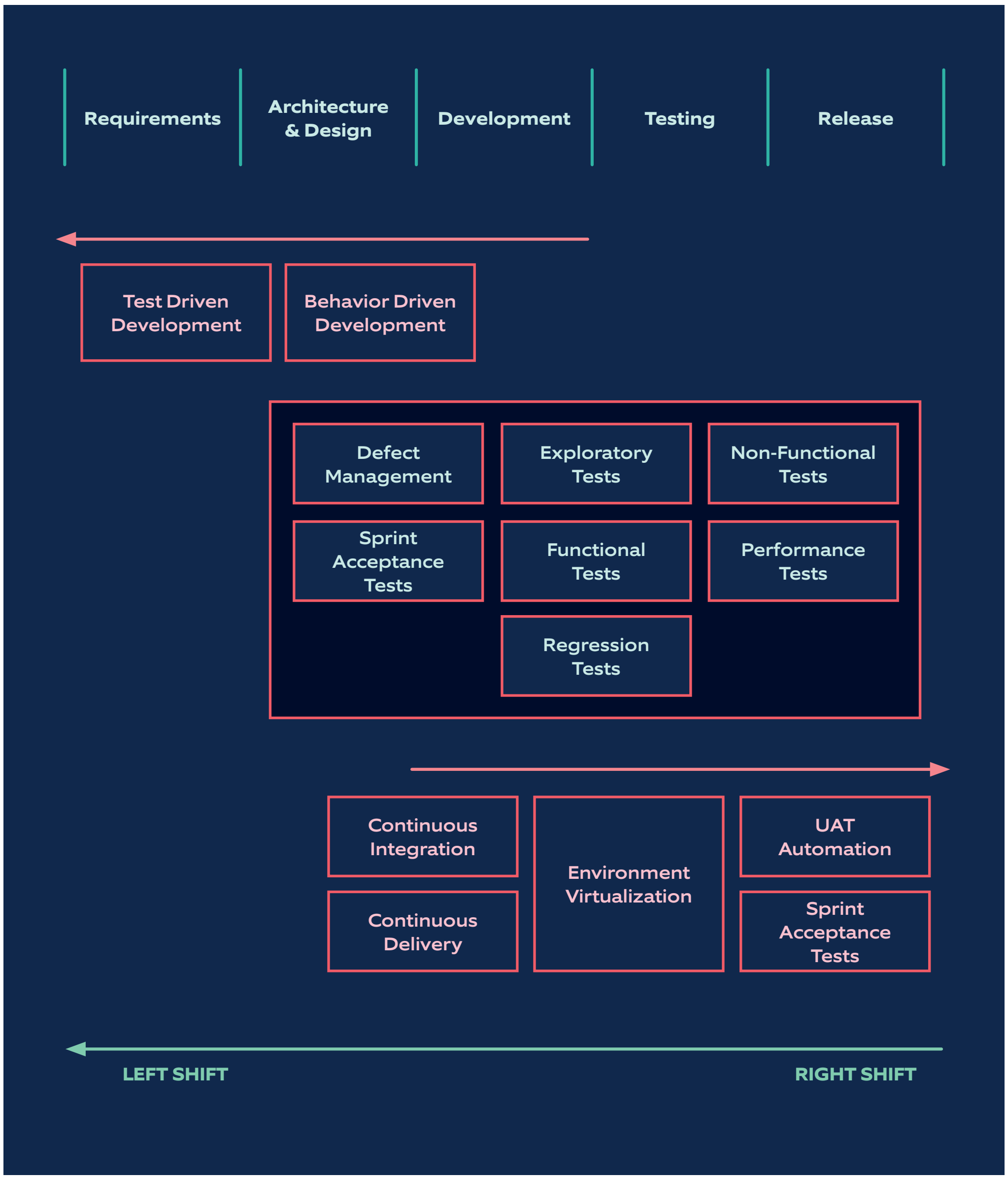
Following is the difference between what Traditional Automation and Cognitive Automation.

Traditional Automation	Cognitive Automation
Failure Detection and Prevention	Failure Prediction
Scripted Automation	Automate the Automation
Limited process automation based on static rules	Inteliegent process automation based on dynamic inference
Individual metrics based on structured data in lifecycle sile	Analytics based on large scale structured and unstructured data across the lifecycle
Limited insight and perpectives	Focus on Insights
Reactive: Deals with “Past”	Predictive: From “What did happen?” to “What could happen?”

The Left Shift

In traditional QA process, we have seen the involvement of QA in the later part of the project when the project has been built and the testing needs to be done just before deployment. This results in poor requirement understanding and poor QA process as now we are validating what has been built and not what was intended to be built. In Cognitive Testing Approach, testing team needs to be involved during the requirement gathering and analysis so that a proper test strategy and a test architecture can be defined for the project. The essence of BDD and API testing is best achieved through this approach. We need to define the Test Lifecycle at the start of the project so that the purpose of agile methodology is not defeated.

ATDD approach also works in similar manner where we write test first and then write code to make them pass. This can only happen in LEFT SHIFT approach. ATDD is like what Unit Testing is in TDD. The idea is to write tests first and then write code to make the tests pass. This approach starts right from the requirement understanding phase of a user story or a ticket. We can define tests which we need to run against the functionality and automate them. Of course, they will fail but eventually pass once we have functionality in place.



Conclusion

What we are aiming is for a Value-Driven Delivery to the customer which involves a holistic view of the things we work on. The idea is to have QA as a part of strategy of the company. When we talk about strategy we talk about goals of the company and how company's portfolio of projects is aligned with the strategy of the company. Managing and monitoring customer experience lies at the heart of the QA. This trickles down to the various aspects how we manage projects, how we develop, test, release and deploy. Creating a Customer-driven Value Proposition should be our aim.
This includes:

1. Minimum viable experience
2. Continuous Delivery
3. Minimum viable quality
4. Minimum viable, validated, value product

References

https://techbeacon.com/how-tech-giants-test-software-theres-no-one-way-qa

http://www.softwaretestinghelp.com/devops-and-software-testing/

https://www.guru99.com/parallel-testing.html

http://www.cigniti.com/blog/how-digital-assurance-is-different-from-traditional-qa/

https://support.smartbear.com/testcomplete/docs/testing-approaches/parallel-testing.html

https://devops.com/7-key-reasons-make-move-sequential-parallel-testing/

https://help.crossbrowsertesting.com/selenium-testing/getting-started/what-is-parallel-testing/

https://www.infosys.com/IT-services/validation-solutions/service-offerings/Documents/api-test-automation.pdf

The Author

Nagarro Inc.

About Nagarro

Nagarro is a global digital engineering leader with a full-service offering, including digital product engineering, digital commerce, customer experience, AI and ML-based solutions, cloud, immersive technologies, IoT solutions, and consulting on next-generation ERP. We help our clients become innovative, digital-first companies through our entrepreneurial and agile mindset, and we deliver on our promise of thinking breakthroughs.

Our guiding principles are defined by one word – CARING, denoting a humanistic, people-first way of thinking with a strong emphasis on ethics. Caring guides us as a global company.

We have a broad and long-standing international customer base, primarily in Europe and North America. This includes many global blue-chip companies, leading independent software vendors (ISVs), other market and industry leaders, and public sector clients.

Today, we are over 18,000 experts across 33 countries, forming a Nation of Nagarrians, ready to help our customers succeed.

www.nagarro.com