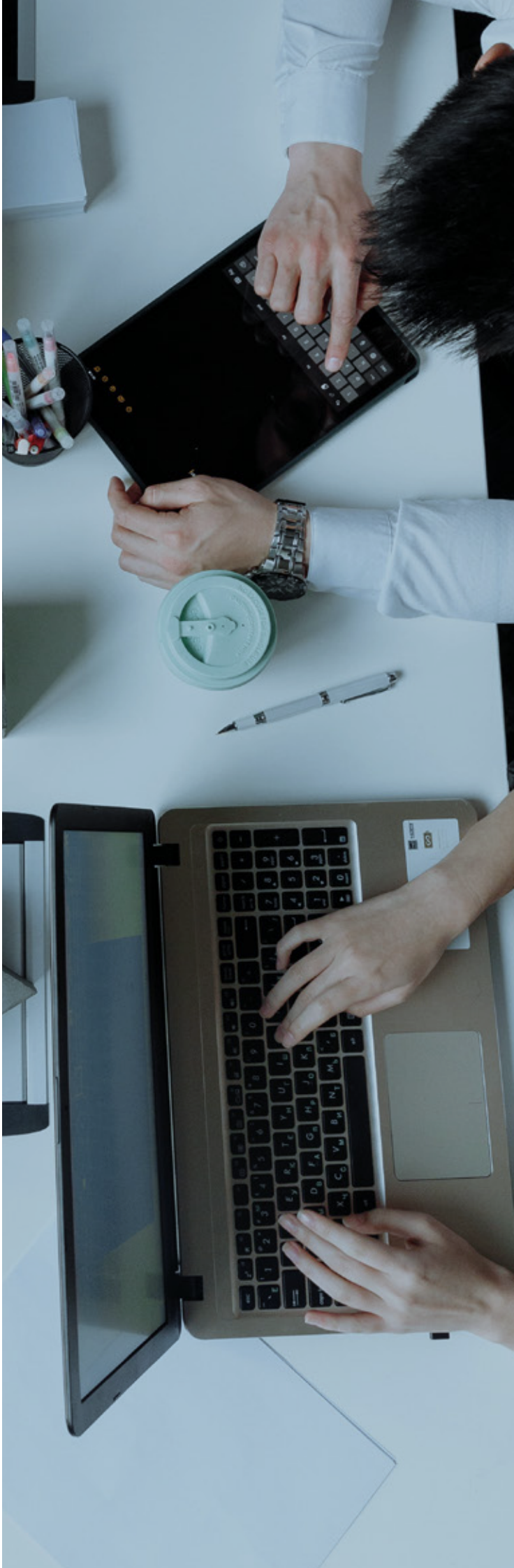
Attribute

LEAWARE

QA in Product Development

Delivering quality

CONTENTS



INTRODUCTION

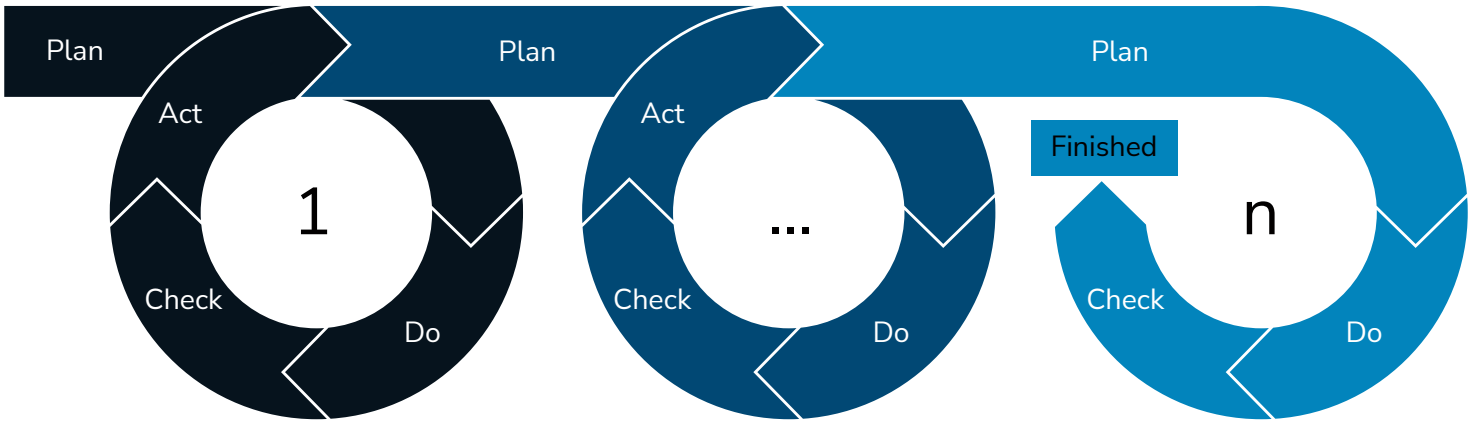
Quality assurance (QA) is a process-centric approach that ensures an organization provides the best possible product or service. It is related to quality control (QC), which focuses on the result, such as testing samples in a batch after production. Although these terms are sometimes used interchangeably, QA focuses on enhancing and improving the process used to create the final result, not the result itself. QA includes planning, design, development, production, and service. At Leaware, we make sure that all our processes go through this important step to ensure that customers receive a high quality product that works well and is free from errors.

Shewhart Cycle

Organizations have access to several QA tools that guide them through steps to ensure their processes are efficient. One of the most popular tools is called the Shewhart cycle – also known as the Deming or PDCA cycle – which was developed by Dr. W. Edwards Deming, a 20th-century American management consultant. The Shewhart QA cycle consists of four steps: Plan, Do, Check, and Act (PDCA). The steps of the Shewhart cycle are repeated to ensure that the process is continually evaluated and improved.

The PDCA Cycle

In the first step (planning) of the PDCA cycle, the organization defines its goals and identifies the process or changes needed to deliver desired results. The second step (do) develops and tests the process or change. The third step (check) monitors and evaluates the process or change to determine whether the results meet the goals. The last step (act) implements the actions needed to achieve the desired improvement. The cycle can then be repeated, starting with planning new goals. At Leaware, we use the PDCA cycle often. Every objective that we have set ourselves and progress is tracked so that when we complete reviews, we can see what actions have been completed to improve the product or service.



Superior Performance in Every Component

The Shewhart cycle is an effective QA method because it analyzes the existing conditions and methods used to deliver products or services to customers. The aim is to ensure excellence in every component of the process. QA helps to determine whether the steps used to provide a product or service are appropriate for the time and conditions. Furthermore, if this cycle is repeated throughout the life cycle of a product or service, it can help increase workers' efficiency as processes are continually improved.

Attention to Detail

QA requires a certain level of detail to be fully implemented in each step. For example, planning can include identifying specific levels of quality or measurable outcomes that the organization wants to achieve. Inspections may involve testing and other objective measures to determine whether goals have been achieved, rather than just a subjective assessment of quality. Taking action may mean overhauling the manufacturing process to correct technical or cosmetic flaws or making small changes to improve efficiency or accuracy.

Competition to provide specialized products and services often leads to breakthroughs, long-term growth, and change. QA verifies that the product offered to the customer is produced and supplied with the best possible materials, in a comprehensive way, and with high standards. QA has the goal of exceeding customer expectations in a measurable and accountable process.

WHAT ARE BUGS?

Bugs have varying impacts on a product's performance, ranging from tiny errors that are easily noticed to serious faults that render the software unusable. All issues should be proactively addressed to provide the best possible user experience. Major issues are usually viewed as urgent and a priority, especially if there's a chance that the customer would want compensation for the error or leave a negative review for the software development company.

Many bugs can affect a product's performance and functionality, but one of the most common impacts is the software crashing. A software crash is when the software stops performing as expected and shuts down, potentially when the user is in the middle of completing an action.

For example, a word processing application could crash while the user is writing an essay or report. This may cause them to lose work if they haven't recently saved their work, which harms productivity and a poor perception of the product.

Typos are also a type of bug. While typos are small coding errors made by a developer, they can create significant errors. A misplaced letter or an incorrect number can cause significant changes to a tool or program's intended functions and interface.

A software bug can disrupt a business's ability to generate leads, interact with users, and facilitate purchases. For example, an e-commerce website may be unable to accept payments from customers. Products might not appear in the customer's cart, or payments could be rejected.

Consumers could become frustrated and look elsewhere, assuming the company is untrustworthy. It is normal to find bugs when developing digital products, but at Leaware we focus on looking for the root cause of them to take them as opportunities for improvement.

Who is responsible for fixing bugs?

Typically, bugs will be discovered by a member of the QA team while running automated tests or by a product manager. If a bug is missed by developers, QA testers, or product managers, it may be identified by a user. This is not ideal, as it would disrupt the user's experience and decrease their satisfaction with the product.

The responsibility for fixing bugs falls on developers, either those who created the product or those who saw issues after initial development. A developer may utilize a debugger to track down bugs and efficiently remove them.

HOW SOFTWARE BUGS CAN BE PREVENTED

The following techniques can help to prevent software bugs:

Test-Driven Development (TDD)

TDD encourages creating failing tests for the product before developing the product. As the development of the product reaches completion, the test cases begin to pass, validating the expected behavior for the product. If tests are created before the product, the likelihood of a product reaching its end-users untested is greatly reduced. TDD can serve as a powerful technique to prevent bugs arising due to inadequate testing.

Continuous Integration Continuous Testing (CICT)

CICT emphasizes that every code change integrated into the central code repository should be automatically tested with predefined test cases. Continuous testing is only possible when automated testing is implemented and integrated with the CI/CD system. To achieve CICT, the test automation system must be integrated with the build system of the product.

Having a CICT pipeline for the product makes sure that any regressions or injections in the product are caught as soon as they are introduced. This saves time and resource that could otherwise be spent searching for changes that caused the regression or injection.

Behavior Driven Development (BDD)

BDD encourages the use of a Domain Specific Language (DSL) between and within teams. Using DSL reduces miscommunication among stakeholders. When BDD is used, tests can be created in a simple text language like English which makes it easier for the team to create and review tests without using complex code syntax.

This flexibility and transparency between teams when creating and reviewing test cases prevent bugs from occurring. Testsigma is a unified test automation tool that has taken test automation to a new level by allowing automation in English. To know more about Testsigma, **Click Here.**

Specification Review and Management

As the scope for the product evolves the product specification also evolves. A dedicated effort may be required to review and track any updates to specifications. Proactively capturing specification updates can help to prevent bugs later in the implementation of the product.

Clear Communication

Open and clear communication among team members can highlight absent or conflicting scenarios in specifications. If every team member is encouraged to communicate about cases or scenarios that are missing/conflict with the specification, issues can be resolved at a much earlier stage.

Encouraging teams to seek input from stakeholders and brainstorm expected behaviors in cases of missing/conflicting specifications can help to reduce bugs arising out of assumptions made for implementation.

One of the main objectives of digital product development management at Leaware is to find a way to reduce the percentage of bugs based on the complexity of the product's programming.

TYPES OF SOFTWARE TESTING

Functional Testing

Functional testing tests business-critical features, functionality, and usability. It ensures that software features and functionality behave as expected without any failures. The entire application is checked against the specifications outlined in the Software Requirements Specification (SRS) document. The types of functional tests include unitary testing, interface testing, and regression testing.

Unit Testing

Unit testing is the first test that developers perform during the development stage. Unit testing tests individual parts/units of a software application at the beginning of the SDLC (Software Development Life cycle). Any function, procedure, method, or module can be unit tested to determine whether it works as expected.

Integration Testing

Integration testing tests different modules of a software application as a group. The software application is made up of various modules that work together. This type of testing identifies errors and problems with the integration of modules.

Non-Functional Testing

Non-functional testing is similar to functional testing. The main difference is that these functions are tested under pressure for observer performance, reliability, usability, and scalability. Non-functional testing, such as load and stress testing, typically uses automation tools and solutions such as LoadView.

Other types of non-functional testing include installation testing, reliability testing, and security testing.

Performance Testing

Performance testing is a type of non-functional testing that is done to determine the speed, stability, and scalability of a software application. The overall goal is to test the performance of an application against systems and network criteria such as CPU usage, page loading speed, traffic peak, and server resource usage. Within performance testing, there are several other types of testing such as load testing and stress testing.

Deep Testing

Deep testing is a type of manual testing where testers thoroughly and rigorously test software. The more features and improvements that are added to the code, the more tests need to be performed to ensure that the overall product works. In addition, it is important to make sure bugs do not reoccur in subsequent releases. Automation is the key to this capability and writing tests will sooner or later become part of your development process. With all that said, is it worth doing manual testing? The short answer is yes and special attention should be paid to deep testing to identify non-obvious errors.

The duration of a deep testing session should not exceed two hours. For deep testing sessions, it is necessary to specify what specific element testers need to focus on.

It is necessary to familiarize all testers with the task, after which they can perform various actions to check the system. This type of testing is inherently costly but is highly effective for identifying user interface issues or testing complex user processes.

Exploratory Testing

Exploratory testing is suitable for specific test scenarios, such as when a user needs to learn about a product or application quickly and provide quick feedback. It helps in assessing the quality of the product from the user's perspective. In many software lifecycles, early iteration is required when teams don't have much time to structure their tests. Exploratory testing is very helpful in this scenario.

Test Case

In software engineering, a test case is a combination of input data, execution conditions, test procedure, and expected results that define a single test. This test achieves a specific software testing goal, such as executing a specific program path or verifying compliance with a specific requirement. Test cases contain data such as:

- Description of a given test case.
- Steps to be performed from starting the application.
- Preconditions or conditions must be met to proceed with the case.
- The expected result and how the application behaves the steps in the test have been completed.

Automated Tests

Automated tests run automatically instead of manually. They consist of programming specific clicks in the application as a real user would do. Testers compare the test against the expected result to see whether the software works as it should.

How These Types of Tests Differ From Each Other

Perhaps you are already aware of the types of testing mentioned above. All tests focus on the reliability and readiness of software applications. However, it is best to explain the differences between these types of tests with examples. In this example, the company has an e-commerce website/app with standard functionality.

To test how the site will perform when many users are using the site at the same time, such as during sales season, load tests will be needed. This will help to detect speed and eliminate potential performance bottlenecks.

To test input and output for each functionality like check-in, login, add to cart, checkout, payment processing, and write to the database, functional testing is needed.

To test the functionality of the cart with checkout and payment module integration, integration testing is needed.

To test whether a newly written module for loading a product is correct and the products are successfully added without any errors or defects, unit testing is needed.

Benefits of These Types of Tests

Performance Testing:

- Assesses the speed and scalability of a website or app.
- Identifies bottlenecks to improve performance.
- Finds errors that are overlooked in functional testing.
- System optimization and function enhancements.
- Ensures the site will be reliable at peak points.

Functional Testing:

- Makes sure the website or app is free of defects.
- Ensures functions behave as expected.
- Ensures that the architecture is correct and has appropriate security.
- Improves the overall quality and functionality.
- Minimizes business risks associated with the website or app.

Integration Testing:

- Ensures that all application modules are well integrated and work as expected.
- Detects interrelated issues and proactively resolves them.
- Checks functionality, reliability, and stability between different modules.
- Detects missed exceptions to improve code quality.
- Supports the CI/CD pipeline.

Unit Testing:

- Detects bugs early in newly developed functionalities or features.
- Minimizes testing costs as problems are identified early.
- Improves code quality with better code refactoring.
- Supports the agile development process.
- Simplifies integration and allows good documentation.

When to Run These Types of Tests

At Leaware, we give you the confidence that each benefit we have listed is closely aligned with how we develop digital products. Performance testing is essential in all development and production environments to make sure the software product works as expected and can handle the expected user load. Functional testing should be done with every build to verify changes and make sure that features match the product's specification.

Integration testing should be done when integrating a new piece of code with another module to make sure there are no conflicts and they work together correctly. Developers complete one-time testing whenever they write code to check the correct input and output.

QUALITY MANAGEMENT

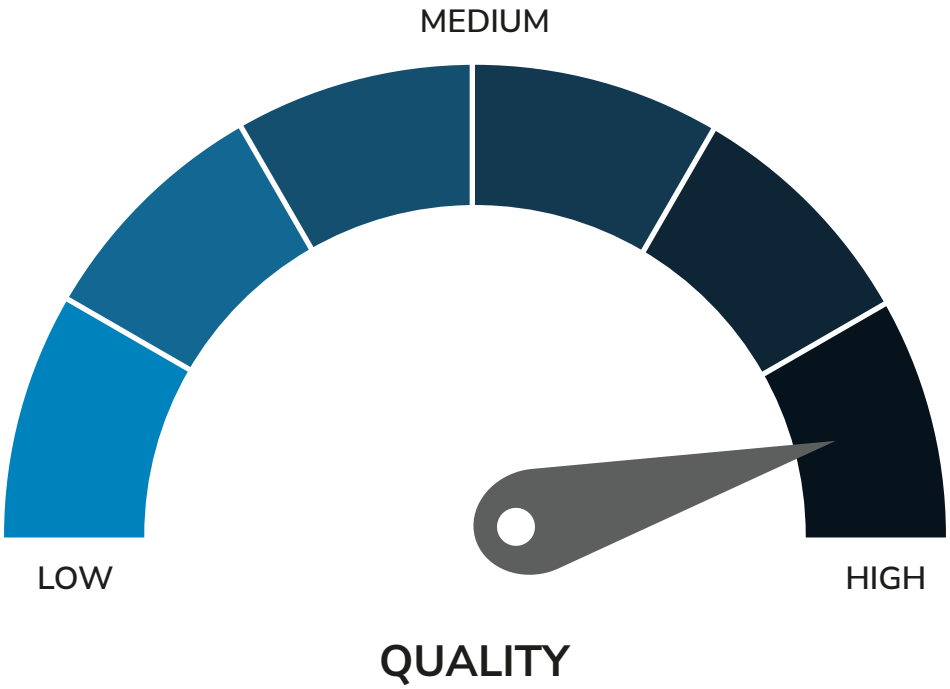
At the present level of development, quality is a complex component, including the quality of the final product, management, delivery or work, and life of users. Quality management is a coordinated and interrelated management activity, built in such a way as to ensure the reliable and uninterrupted operation of the organization.

Organizational management – in the context of quality – means that all activities are reaching benchmarks. To meet these benchmarks, the organization needs to have a developed system of plans, necessary resources, and take action to achieve goals.

Quality Management of the Project

Quality management at Leaware is a support process with a significant impact on management decisions.

To develop a quality management for information systems implementation projects, it is necessary to compile procedures for the regulations. One of the main components of project management is to prevent loss of value by reducing the quality of the product or service. Companies that provide services for the implementation of information systems accumulate knowledge about emerging problems and attempt to prevent future problems.



Quality management has four main components:

- Quality control.
- Quality assurance.
- Quality planning.
- Quality improvement.

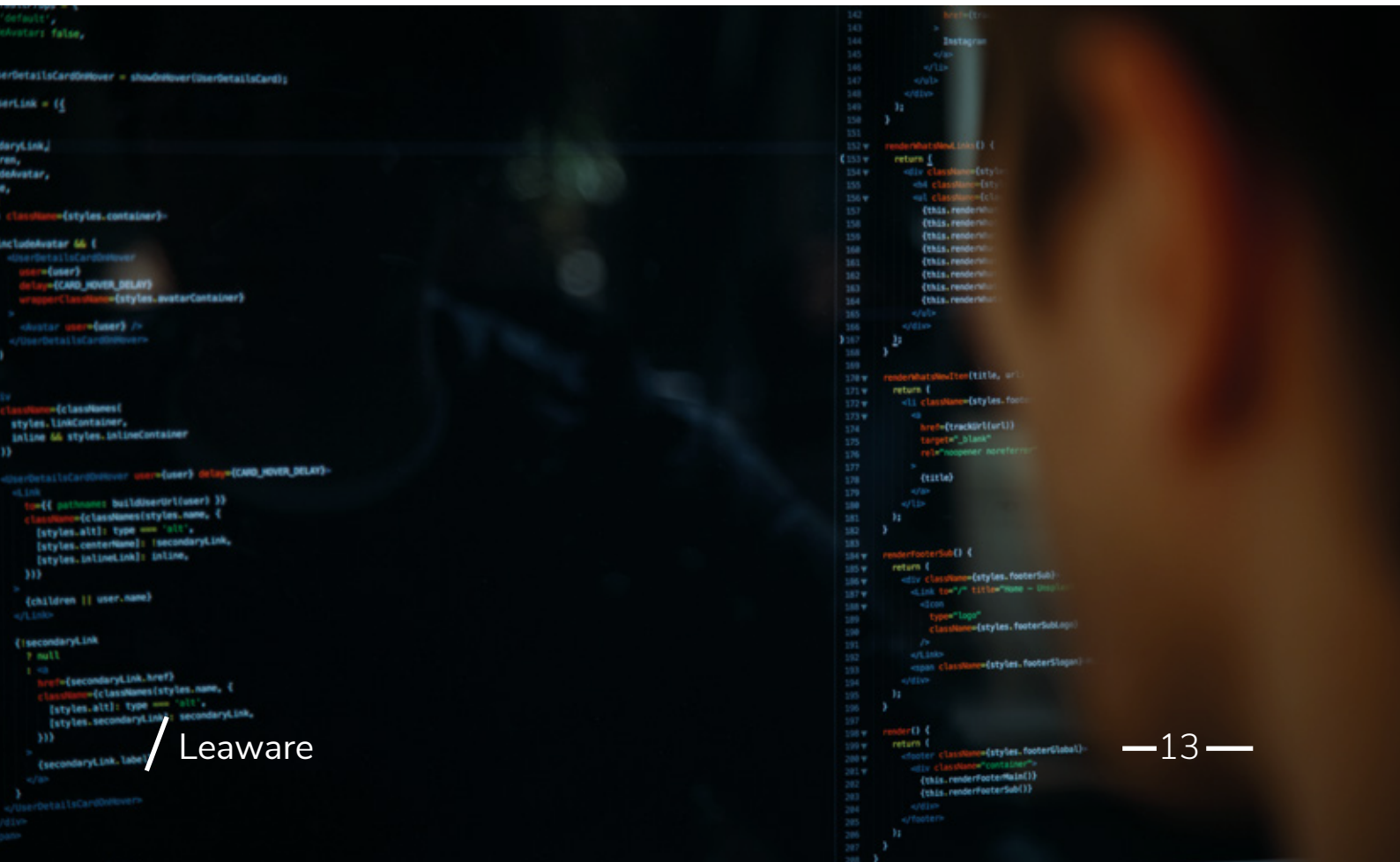
QC assesses whether the object of control meets requirements. Evaluation activities may include measurements, tests, observations, monitoring, verification, calibration, and other activities that compare the values of observed characteristics with specified ones.

QA is a systematic activity that fulfills requirements. It includes production, management, logistics, and maintenance.

Quality planning ascertains the necessary characteristics of an object and sets target values. Quality planning also determines the processes and resources needed to achieve objectives.

Quality improvement is an action that improves the organization's ability to meet requirements. Fixed improvement considers products, processes, management systems, and the organization.

Quality management is a voluminous section of applied science, which contains the philosophy of quality management theory and practical methods.



QUALITY CONTROL AND TESTING

Testing

Testing is the initial level of control. During the testing process, the software is checked against the brief to make sure that it meets the requirements. In this process, the software is checked for compliance, testers make sure that the software meets the criteria, all defects identified are resolved, and the product is re-checked to ensure that all inconsistencies are removed. Testing is not just clicking the mouse on a certain plan. To complete the testing process, the company will need to have an automated testing process or an experienced developer. The main goal of this process is to find as many defects as possible and identify why they have occurred. At Leaware we have 10+ years of experience completing this process for our clients.

Quality Control

There is testing at this stage, but QC is not only about that. Similarly to the first stage, this process checks compliance and assesses the quality of the product to determine whether it's ready for release.

To do this, the priority and number of errors are evaluated. If critical defects remain in the software, the QC specialist will send the product back to the testing process so the errors can be removed. The main goal of this process is to assess the overall quality of the product.

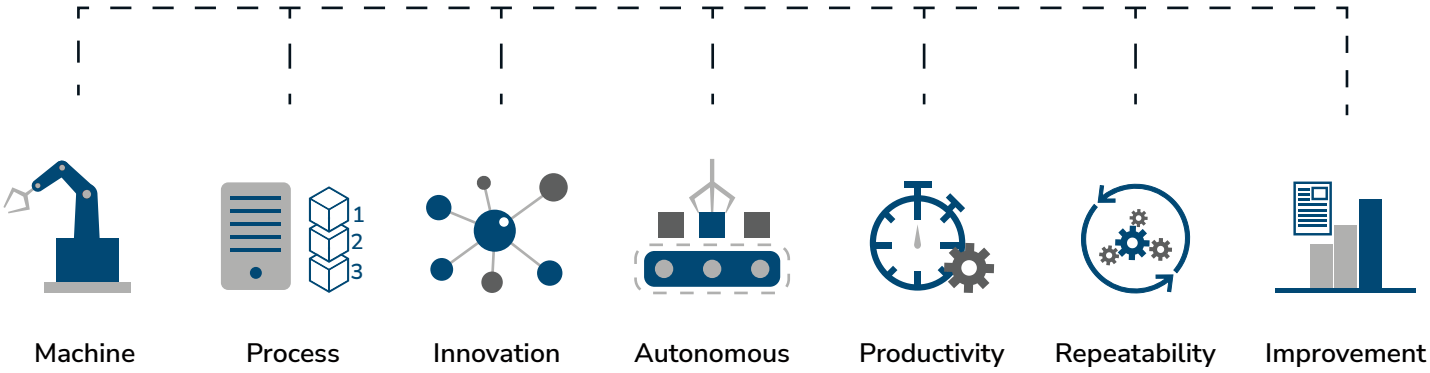
Quality Assurance

QA specialists monitor the quality of the tester's work and the entire development process. The underlying goal is to create a system that will minimize the number of errors and facilitate testing. For example, this could mean introducing an improved scheduling system, so that both developers and testers have enough time to do quality work. QA may include checking the code for compliance against existing standards, checking documents, and implementing new QC methods.

TEST AUTOMATION

Test automation is a term used in software testing, other IT-related testing, and QA. Test automation generates tests without human input. Different types of test automation help businesses to achieve goals such as software testing with fewer resources which makes the testing process more efficient.

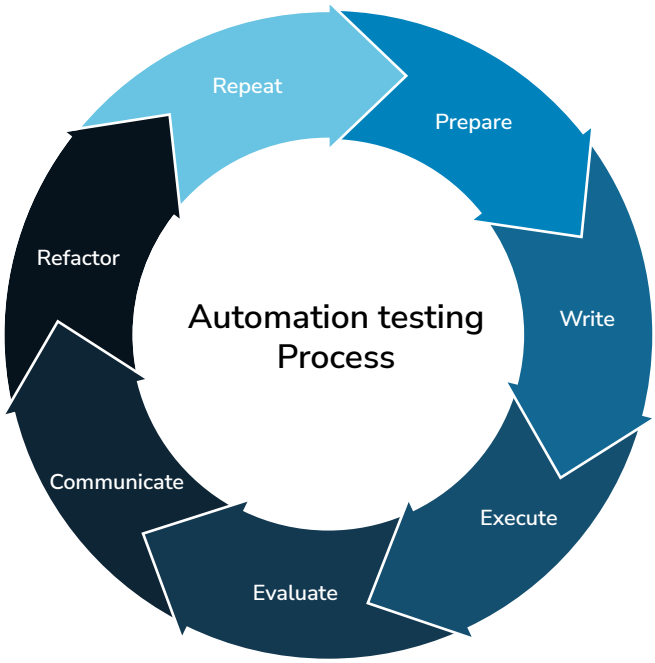
In addition to test automation, certain kinds of innovations help make software testing easier and more efficient like scriptless test automation. This involves a system where programmers don't need to write scripts for each test case. By using reusable code assets, scriptless test automation resources can help companies test software in a less labor-intensive way.



Automated Testing Process

Before carrying out automated testing tasks, it is necessary to understand and master the process of implementing automated testing. Without mastering the automated testing process, you will not be able to carry out automated testing tasks smoothly. The test process for automated testing and functional testing are similar. The content of this article is summed up based on years of the author's automated testing experience.

After receiving a project, Leaware first analyze which modules of the system are suitable for automated testing, understand the significance of automated testing implementation, and the value it can bring. If automated testing is carried out blindly and without purpose, results are likely to be meaningless.



Test Scheme Selection

Test scheme selection is the selection of test tools. When carrying out automated testing tasks, you must first choose which implementation method to use. The most common method is tool recording or encoding and the author recommends the latter. More developer-defined functions can be implemented through coding, especially when there are many test cases, maintenance, test data management, and other issues that should be considered.

At present, the more mainstream automated testing frameworks are Selenium, Robot Framework, Appium, Roboium, and MonkeyRunner. When selecting the test plan, the development language, project architecture, and project environment used by the project team should be considered.

Preparation of the Test Environment

Before automated testing can be performed, the test environment needs to be prepared. The test environment generally includes a tool installation environment and an automated test environment. If you use Selenium and Appium to carry out automated testing, you must install a language environment, such as Python or Java. It is also important to consider continuous integration environments and version management at this point.

Test Framework Design

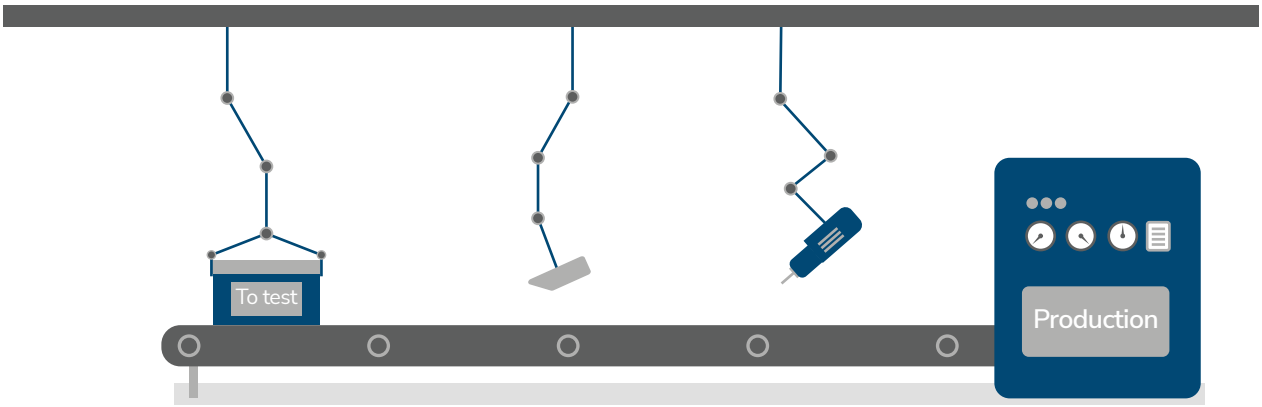
When many automated test cases are written, completing maintenance on the test cases becomes difficult. Common elements used in test cases, such as test data, configuration files, and log files need to be stored separately. This is because in the development, the test project is often completed by a team rather than an individual.

Test Case Execution

After the automated test case is executed, workers can learn how many test cases were executed in this test, the number of passes and failures, and the reasons for any failures from the test report. They can also build automation tasks at regular intervals through Jenkins. If a code submission or an update is needed, the automatic build automation task will trigger, generating test results that are sent to the relevant person.

Test Automation in a Productive Environment

Experienced IT professionals know firsthand that even a very well-tested product may not work in real-life settings. Therefore, it is good practice to run existing regression tests in a productive environment. The CRUD (Create, Read, Update, and Delete) method can be used to divide auto-tests into groups. In a productive environment, it is often only possible to run only R tests that do not change any data. Despite this limitation, automated testing in a productive environment is a critical step in a product's release.



Which Test Cases Should be Automated?

To increase the automation ROI, test cases for automation can be selected based on the following criteria:

- High risks and failures are unacceptable, which is extremely important for sectors such as banking.
- Test scenarios are regularly repeated.
- Test scripts are very complex and tedious to execute manually.
- Time-consuming test cases.

When Not to Use Automation

The following are examples of when it would be inappropriate to use automation for test cases:

- New test cases that have not been run manually at least once.
- Test scenarios where the requirements change frequently.
- Test cases that are run on an ad hoc basis.

Maintenance of Automated Testing

This stage of automated testing is carried out to check whether new features added to the software work. Automated testing maintenance occurs when new automation scripts need to be tested to maintain and improve the effectiveness of the automation scripts with each successive release cycle.

Best Practice for Effective Test Automation

To maximize the ROI on your automation investment, follow these guidelines:

- The scope of automation needs to be defined in detail before the start of the project. This makes sure that expectations are met.

- Determine the right automation tool. The tool should not be chosen based on its popularity. Instead, it should match the automation requirements of the project that is being worked on.

- Choose the right framework.

- Scripting standards must be followed when writing scripts for automation. Some of the main standards are:

-Create unified scripts, comments, and code indents.

-Develop rules for naming test scenarios.

-Attach the necessary documents if, for example, it is difficult to understand the passage of a test scenario without a screenshot and/or specification.

-Define metrics and track them. The success of automation cannot be determined only by comparing the effort expended on another type of testing.

-The main metrics are the percentage of detected defects, the time required to test the release automation, minimum time required for release, customer satisfaction index, and performance improvement.

CONCLUSION

Performance testing is essential for all development and production environments to ensure your website or application is up to speed and can handle the expected user load. Functional testing should be done with every build to verify changes and to make sure that features match the specifications and requirements. Integration testing should be done when integrating a new piece of code to make sure there are no conflicts. One-time testing should be done by developers whenever they write any code to check the correct input and output. At Leaware, we fervently believe that QC and management in automated processes had a positive impact on the creation, design, and development of digital products.

ABOUT THE AUTHORS

Tom Soroka
Leaware Founder
Business Development Manager

Damian Wasilewski
Project Manager
Business Development Manager

Carlos Lopes
Marketing Specialist
Business Development Manager

Since 2010, LEAWARE has been helping start-ups and already established companies build and develop their digital solutions, making them grow, boost their businesses, and succeed.

LEAWARE

Contact us
ask@leaware.com
Visit our website
www.leaware.com
Or see our Clutch profile
Clutch

Follow us on social media

Thanks for reading!

To receive more of our "QA in Product Development" scan the QR code below.

We build the right software.
We build the software right.

LEAWARE