altexsoft
software r&d engineering

WHITEPAPER

Quality Assurance, Quality Control and Testing — **the Basics of Software Quality Management**

Introduction

When you buy a pear, you can instantly evaluate its quality: the size and shape, ripeness, the absence of visible bruising. But only as you take the first bite, will you be able to see if the pear is really that good. Even an extremely good-looking pear might taste sour or have a worm in it.

The same applies to almost any product, be it a physical object or a piece of software. A website you find on the internet might seem fine at first, but as you scroll down, go to another page, or try to send a contact request, it can start showing some design flaws and errors.
This makes quality control so important in every field, where an end-user product is created. Yet, a sour pear won't cause as much damage as a self-driving car with poor quality autopilot software. A single error in an EHR system might put a patient's life at risk, while an eCommerce website that has performance issues might cost the owner millions of dollars in revenue.

That is why we at AltexSoft put a premium on the quality of software we build for our clients. In this paper we will share our insights on the quality assurance and testing process, our best practices and preferred strategies.

1. The Concept of Software Quality: Quality Assurance (QA), Quality Control (QC) and Testing

While to err is human, sometimes the cost of a mistake might be just too high. History knows many examples of situations when software flaws have caused billions of dollars in waste or even lead to casualties: from Starbucks coffee shops being forced to give away free drinks because of a register malfunction, to the F-35 military aircraft being unable to detect the targets correctly because of a radar failure.

In order to make sure the released software is safe and functions as expected, the concept of software quality was introduced. It is often defined as *"the degree of conformance to explicit or implicit requirements and expectations"*. These so-called explicit and implicit expectations correspond to the two basic levels of software quality:

- **Functional** — the product's compliance with functional (explicit) requirements and design specifications. This aspect focuses on the practical use of software, from the point of view of the user: its features, performance, ease of use, absence of defects.
- **Non-Functional** — system's inner characteristics and architecture, i.e. structural (implicit) requirements. This includes the code maintainability, understandability, efficiency and security[2].

The structural quality of the software is usually hard to manage: It relies mostly on the expertise of the engineering team and can be assured through code review, analysis and refactoring. At the same time, functional aspect can be assured through a set of dedicated **quality management** activities, which includes quality assurance, quality control and testing.

Often used interchangeably, the three terms refer to slightly different aspects of software quality management. Despite a common goal of delivering a product of the best possible quality, both structurally and functionally, they use different approaches to this task.

QA, QC and Testing in software development process

Quality Assurance is a broad term, explained on the Google Testing Blog as *"the continuous and consistent improvement and maintenance of process that enables the QC job"*. As follows from the definition, QA focuses more on organizational aspects of the quality management, monitoring the consistency of the production process.

Through **Quality Control** the team verifies the product's compliance with the functional requirements. As defined by Investopedia, it is a *"process through which a business seeks to ensure that product quality is maintained or improved and manufacturing errors are reduced or eliminated"*. This activity is applied to the finished product and performed before the product release.

In terms of manufacturing industry, it is similar to pulling a random item from an assembly line to see if it complies with the technical specs.

Testing is the basic activity aimed at detecting and solving technical issues in the software source code and assessing the overall product usability, performance, security and compatibility. It has a very narrow focus and is performed by the test engineers in parallel with the development process or at the dedicated testing stage (depending on the methodological approach to the software development cycle).

	QA	QC	Testing
Purpose	Setting up adequate processes, introducing the standards of quality to prevent the errors and flaws in the product	Making sure that the product corresponds to the requirements and specs before it is released	Detecting and solving software errors and flaws
Focus	Processes	Product as a whole	Source code and design
What	Prevention	Verification	Detection
Who	The team including the stakeholders	The team	Test Engineers, Developers
When	Throughout the process	Before the release	At the testing stage or along with the development process

The concepts of quality assurance, quality control, and testing compared

If applied to the process of car manufacturing, having a proper **quality assurance process** means that every team member understands the requirements and performs his/her work according to the commonly accepted guidelines. Namely, it is used to make sure that every single action is performed in the right order, every detail is properly implemented and the overall processes are consistent, so that nothing can cause negative impact on the end product.

Quality control can be compared to having a senior manager walk into a production department and pick a random car for an examination and test drive. **Testing activities**, in this case, refer to the process of checking every joint, every mechanism separately, as well as the whole product, whether manually or automatically, conducting crush tests, performance tests and actual or simulated test drives.

Due to its hands-on approach, software testing activities remain a subject of heated discussion. That is why we will focus primarily on this aspect of software quality management in this paper. But before we get into the details, let's define the main principles of software testing.

2. The Main Principles of Software Testing

Formulated over the past 40 years, the seven principles of software testing represent the ground rules for the process. These are:

Testing shows presence of mistakes	Testing is aimed at detecting the defects within a piece of software. But no matter how thoroughly the product is tested, we can never be 100 percent sure that there are no defects. We can only use testing to reduce the number of unfound issues.
Exhaustive testing is impossible	There is no way to test all combinations of data inputs, scenarios and preconditions within an application. For example, if a single app screen contains 10 input fields with 3 possible value options each, this means to cover all possible combinations, test engineers would need to create 59,049 (310) test scenarios. And what if the app contains 50+ of such screens? In order not to spend weeks creating millions of such less possible scenarios, it is better to focus on potentially more significant ones.
Early testing	As mentioned above, the cost of an error grows exponentially throughout the stages of the SDLC. Therefore it is important to start testing the software as soon as possible, so that the detected issues are resolved and do not snowball.
Defect clustering	This principle is often referred to as an application of the Pareto Principle to software testing. This means that approximately 80 percent of all errors are usually found in only 20 percent of the system modules. Therefore, if a defect is found in a particular module of a software program, the chances are there might be other defects. That is why it makes sense to test that area of the product thoroughly.

Pesticide paradox	Running the same set of tests again and again won't help you find more issues. As soon as the detected errors are fixed, these test scenarios become useless. Therefore, it is important to review and update the tests regularly in order to adapt and potentially find more errors.
Testing is context dependent	Depending on their purpose or industry, different applications should be tested differently. While safety could be of primary importance for a fintech product, it is less important for a corporate website. The latter, in its turn, puts an emphasis on usability and speed.
Absence-of-errors fallacy	The complete absence of errors in your product does not necessarily mean its success. No matter how much time you have spent polishing your code or improving the functionality, if your product is not useful or does not meet the user expectations it won't be adopted by the target audience.

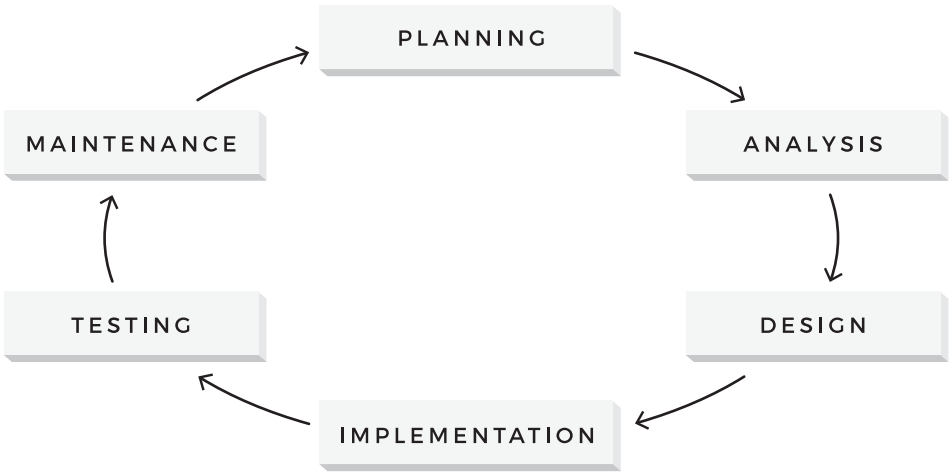
While the above-listed principles are undisputed guidelines for every software testing professional, there are more aspects to consider. Some sources note other principles in addition to the basic ones:

- Testing must be an independent process handled by unbiased professionals.
- Test for invalid and unexpected input values as well as valid and expected ones.
- Testing should be performed only on a static piece of software (no changes should be made in the process of testing).
- Use exhaustive and comprehensive documentation to define the expected test results.

3. The Role of Testing in Software Development Life Cycle

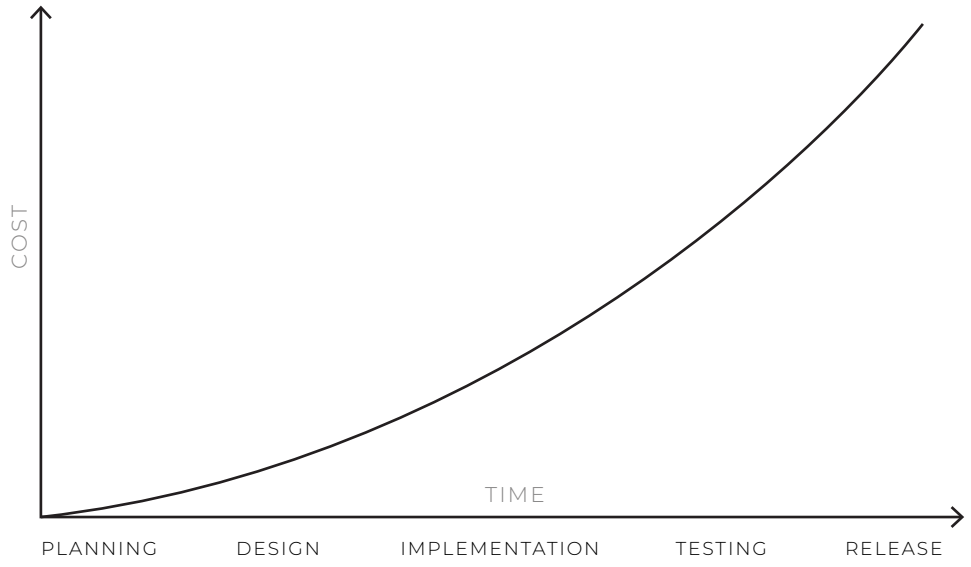
3.1. Waterfall Model

Representing a traditional software development life cycle includes, the Waterfall model **6 consecutive phases**: planning, analysis, design, implementation, testing and maintenance.



Software Development Life Cycle

In the testing phase a product, that is already designed and coded, is being thoroughly tested before the release. However, the practice shows that software errors and defects detected at this stage might be too expensive to fix, as the cost of an error tends to increase throughout the software development process.

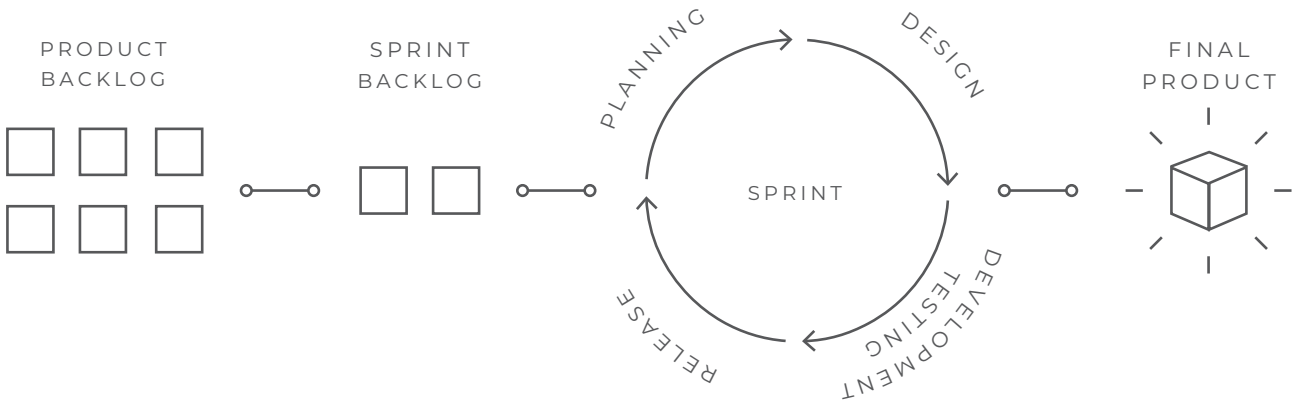


The cost of an error throughout the SDLC

For example, if there is an error in the specifications, detecting it early in the planning stage wouldn't cause significant losses to your business. However, the damage grows exponentially throughout the further stages of the process. If such an error is detected at the design stage, you will need to rework your designs to fix it. But if you aren't able to detect the mistake before the product is built, you might need to make some major changes to the design as well as the source code. This will require a significant amount of effort and investment.

The same is the case for errors produced in the process of implementation. If a feature has a flaw in its logic, building more functionality on top of it might cause a serious damage in the long run. Therefore, it is better to test every feature while the product is still being built. This is where iterative agile methods prove beneficial.

3.2. Agile Testing



Agile Development Cycle

Being an integral part of the software development process, **Agile** breaks the development process into smaller parts, iterations and sprints. This allows testers to work in parallel with the rest of the team throughout the process and fix the flaws and errors immediately after they occur.

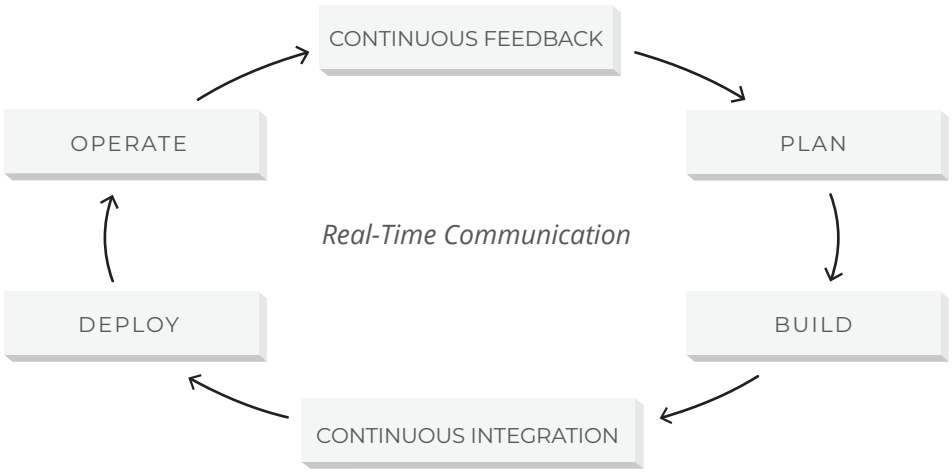
The main purpose of such process is to deliver new software features fast and with the best quality. Therefore, this approach is less cost-intensive: Fixing the errors early in the development process, before more problems snowball, is significantly cheaper and requires less effort. Moreover, efficient communication within the team and active involvement of the stakeholders speeds up the process and allows for better informed decisions.

The Agile testing approach is more about building up a QA practice as opposed to having a QA team. Amir Ghahrai, a Senior Test Consultant at Amido, comments on this matter:

> *"By constructing a QA team, we fall in the danger of separating the testers from vital conversations with the product owners, developers, etc. In Agile projects, QA should be embedded in the scrum teams because testing and quality is not an afterthought. Quality should be baked in right from the start."*

3.3. DevOps Testing

For those who have Agile experience, DevOps gradually becomes a common practice. This new software development methodology requires a high level of coordination between various functions of the deliverable chain, namely development, QA, and operations.



A DevOps lifecycle

DevOps is often referred to as an extension of Agile that bridges the gap between development along with QA and operations. However, unlike Agile, DevOps includes the concept of continuous development where the code, written and committed to version control, will be built, deployed, tested and installed in the production environment that is ready to be consumed by the end-user. DevOps places a great emphasis on automation and continuous integration tools that allow for the high-velocity delivery of applications and services.
The fact that testing takes place at each stage in the DevOps model changes the role of testers and the overall idea of testing. Therefore, to be able to effectively carry out testing activities, testers are now expected to have technical skills and even be code savvy.

According to the PractiTest survey, the Agile trend is an undisputed leader, while almost 90 percent of respondents work at least in some Agile projects within their organizations. That said, a third of the respondents is still applying the Waterfall model in some projects, following a steady decrease in the use of that method. DevOps keeps growing, just slower than before.

4. The Process of Software Testing in Practice

Organizing a software testing process can be quite challenging. We at AltexSoft follow the three major steps in software testing process: planning, execution, and reporting.

The stages of software testing

4.1. Test Planning: Artifacts and Strategy

As any other formal process, testing activities are typically preceded by thorough preparations and planning. The main goal of this stage is to make sure the team understands the customer objectives, the main purpose of the product, the possible risks they need to address, and the outcomes they expect to achieve. One of the documents created at this stage, the **mission or assignment of testing**, serves to solve this task. It helps align the testing activities with the overall purpose of the product and coordinates the testing effort with the rest of the team's work.

Roger S. Pressman, a professional software engineer, famous author and consultant, states:

> *"**Strategy for software testing** provides a roadmap that describes the steps to be conducted as part of testing, when these steps are planned and then undertaken, and how much effort, time, and resources will be required."*

Also referred to as test approach or architecture, test strategy is another artifact of the planning stage. James Bach, a testing guru who created the Rapid Software Testing course, identifies the purpose of a test strategy as *"to clarify the major tasks and challenges of the test project."* A good test strategy, in his opinion, is product specific, practical and justified.

Depending on when exactly in the process they are used, the strategies can be classified as preventive or reactive. In addition to that, there are several types of strategies, that can be used separately or in conjunction:

Strategy	Character	Primary Focus Area	Use Case
Analytical	Preventive	The strategy focuses on risk and requirements analysis to create a basis for planning, building and estimating the tests.	When building a well-defined product from scratch.
Model-Based	Preventive	The system is tested in accordance with the predefined model and should completely correspond to it in order to be considered valid.	When using an existing product as a basis for a new one, or enhancing the legacy system.
Methodical	Preventive	This strategy adheres to a custom pre-planned, systematic approach.	Often used in heavily-regulated industries, to build a product that complies with the requirements.

Strategy	Character	Primary Focus Area	Use Case
Process/ Standard-Compliant	Preventive	Unlike the previous one, this strategy relies on a standard strategy, with little or no adaptation.	Used by the teams that lack experience or time for building a custom testing approach.
Dynamic	Reactive	It prioritizes finding as many errors and defects as possible (the attack-based approach and the exploratory approach).	When there is a need to find and fix the issues with minimum time and effort.
Consultative/ Directed	Reactive	It relies on the users or developers to define the areas of testing or even to handle the tests themselves.	Applied to domain-specific products, that require additional expert guidance.
Regression-averse	Reactive	This strategy prioritizes the automation of functional tests either before the release or after.	Best used with live, well-established products.

The seven types of test strategies

While a test strategy is a high-level document, **test plan** has a more hands-on approach, describing in detail what to test, how to test, when to test and who will do the test. Unlike the static strategy document, that refers to a project as a whole, test plan covers every testing phase separately and is frequently updated by the project manager throughout the process.

According to the IEEE standard for software test documentation, a test plan document should contain the following information:

- Test plan identifier
- Introduction
- References (list of related documents)
- Test items (the product and its versions)
- Features to be tested
- Features not to be tested
- Item pass or fail criteria
- Test approach (testing levels, types, techniques)
- Suspension criteria
- Deliverables (Test Plan (this document itself), Test Cases, Test Scripts, Defect/Enhancement Logs, Test Reports)
- Test environment (hardware, software, tools)
- Estimates
- Schedule
- Staffing and training needs
- Responsibilities
- Risks
- Assumptions and Dependencies
- Approvals

Writing a plan, which includes all of the listed information, is a time-consuming task. In agile methodologies, with their focus on the product instead of documents, such a waste of time seems insufficient.

To solve this problem, James Whittaker, a Technical Evangelist at Microsoft and former Engineering Director at Google, introduced The 10 Minute Test Plan approach. The main idea behind the concept is to focus on the essentials first, cutting all the fluff by using simple lists and tables instead of large paragraphs of detailed descriptions. While the 10-minute time-box seems a little bit unrealistic (None of the teams in the original experiment was able to meet this requirement), the idea of reducing and limiting the planning time itself is highly reasonable. As a result, 80 percent of the planning can be finished within only 30 minutes.

4.2. Design and Execution

As a starting point for the test execution, we need to define what is subject to testing. In order to answer this question, QA teams develop test cases. In a nutshell, a **test case** describes the preconditions, desired outcomes and postconditions of a specific test scenario, aimed at verifying that a feature meets the basic requirements.

The next step in test execution is **setting up the testing environment**. The main criteria for this part is to make sure that the testing environment is as close to the end user's actual environment (hardware and software) as possible. For example, a typical test environment for a web application should include Web Server, database, OS, and browser.

The software testing process identifies two broad categories: static testing and dynamic testing.

Static testing initially examines the source code and software project documents to catch and prevent defects early in the software testing life cycle. Also called non-execution technique or verification testing, static testing could be performed as inspections, informal and technical reviews, or reviews during walkthrough meetings. Informal review is a cheap testing variant that a QA analyst can conduct anytime during the project. Inspection, also called a formal review, is planned and controlled by the moderator. During the review meeting, errors found by QA analysts are discussed and documented in the review report.

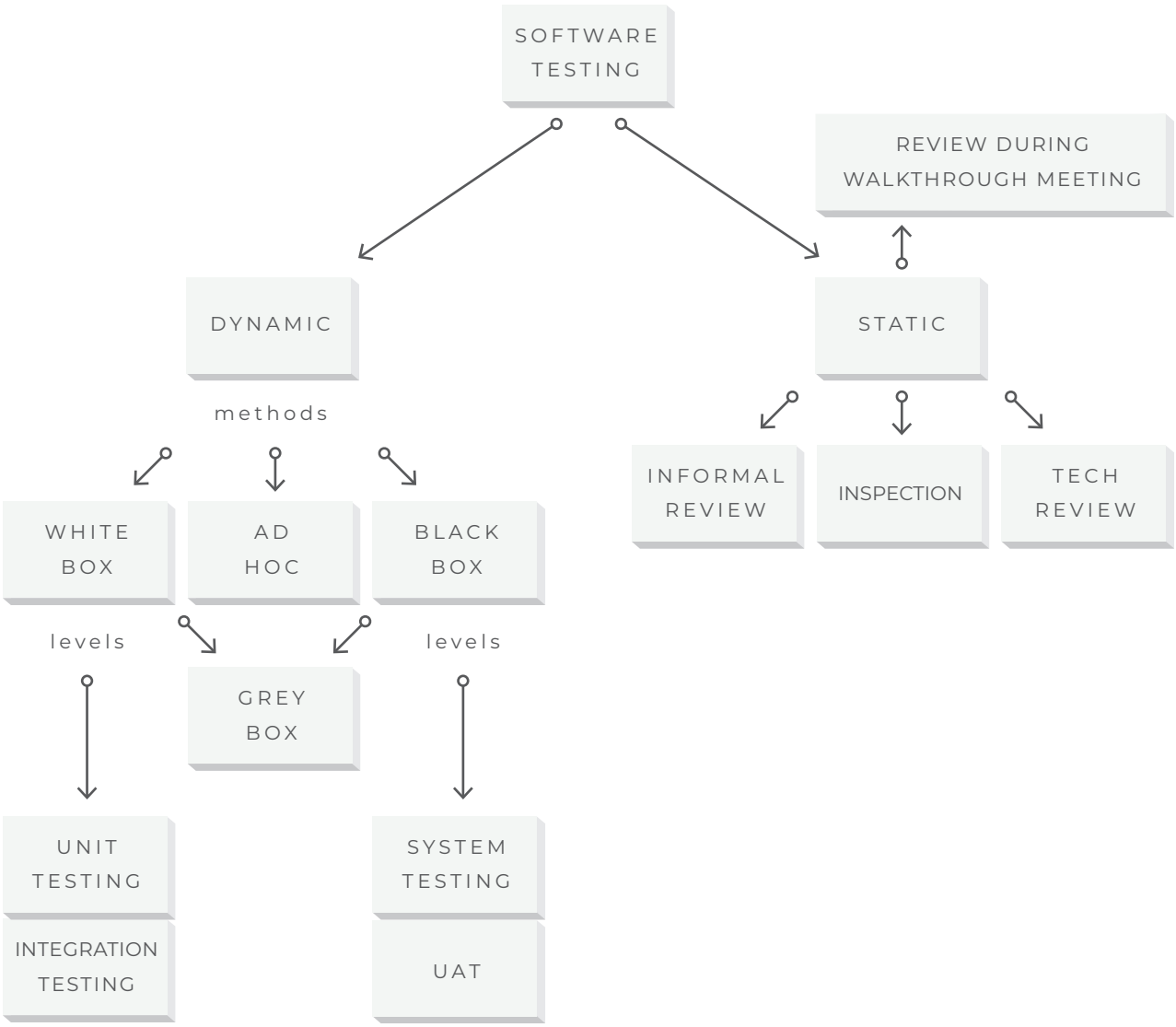
As soon as the primary preparations are finished, the team proceeds with **dynamic testing** where software is tested during execution. This whitepaper has the most focus on the dynamic testing process as a practical and most commonly used way to validate code behavior. Dynamic testing can be described by methods, levels, and types of underlying QA activities. Let's have a closer look at this segment of the dynamic testing process.

Software testing methods are the ways the tests are conducted. They include *black box testing, white box testing, grey box testing*, and *ad hoc testing*.

Software testing levels describe stages of software development when testing is conducted. That said, there are four progressive testing levels based on the area they focus on the software development process: *unit testing, integration testing, system testing,* and *user acceptance testing (UAT).*

Software testing types are the approaches and techniques that are applied at a given level using an appropriate method to address the test requirements in the most efficient manner. They are vast in number while serving different objectives.

To sum up, you can do *use case testing* (a type) during *system* or *acceptance testing* (a level) using *black box testing* (a method).



The software testing process division: static and dynamic testing

4.3. Documentation and Reporting

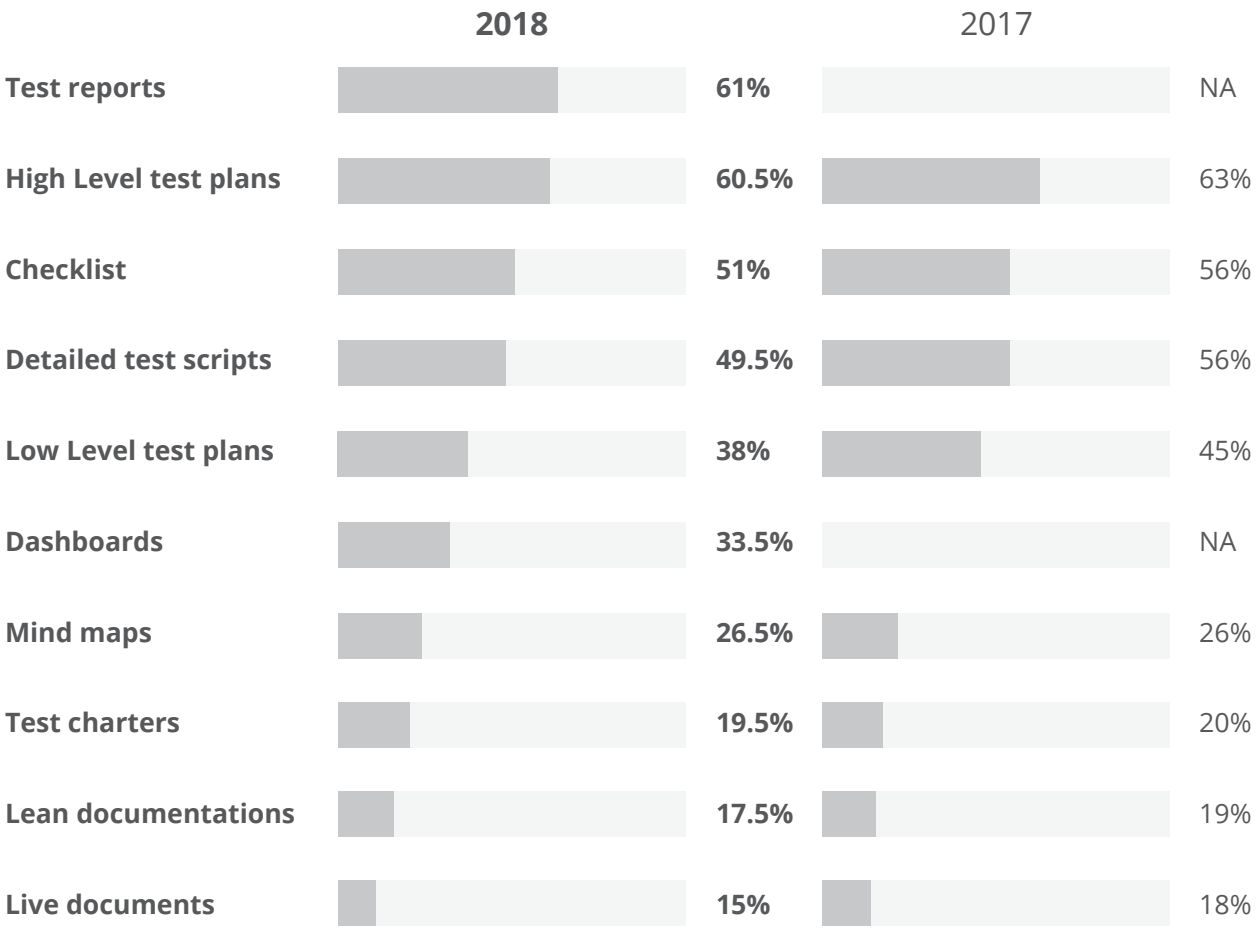
As there is no perfect software, the testing is never 100 percent complete. It is an ongoing process. However, there exists the so-called "exit criteria", which defines whether there was "enough testing" conducted, based on the risk assessment of the project.

There are common points that are present mostly in exit criteria:

- Test case execution is 100 percent complete.
- A system has no high priority defects.
- Performance of the system is stable regardless of the introduction of new features.
- The software supports all necessary platforms and/or browser
- User acceptance testing is completed.

As soon as all of these criteria (or any custom criteria that you have set in your project) are met, the testing comes to its closure.

The testing logs and status reports are documented throughout the process of the test execution. Every issue found in the product should be reported and handled accordingly. The test summary and test closure reports are prepared and provided to the stakeholders. The team holds a retrospective meeting in order to define and document the issues that occurred during the development and improve the process.

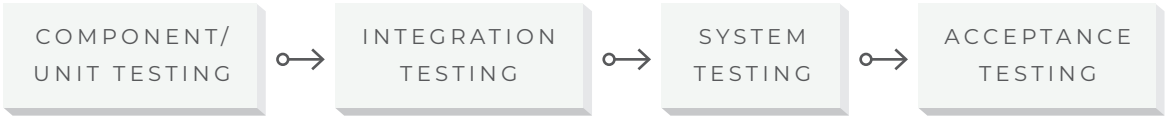


PractiTest Testing documentation survey. From the STATE OF TESTING REPORT 2018

According to the survey conducted by PractiTest, an end-to-end QA and test management solution, there is a constant decrease in the amount of formal testing documentation written. This tendency signals the need to streamline testing all across the industry.

5. The Levels of Testing

A piece of software is more than several lines of code. It is usually a multilayer, complex system, incorporating dozens of separate functional components and third-party integrations. Therefore, efficient software testing should go far beyond just finding errors in the source code. Typically, the testing covers the following levels of software.



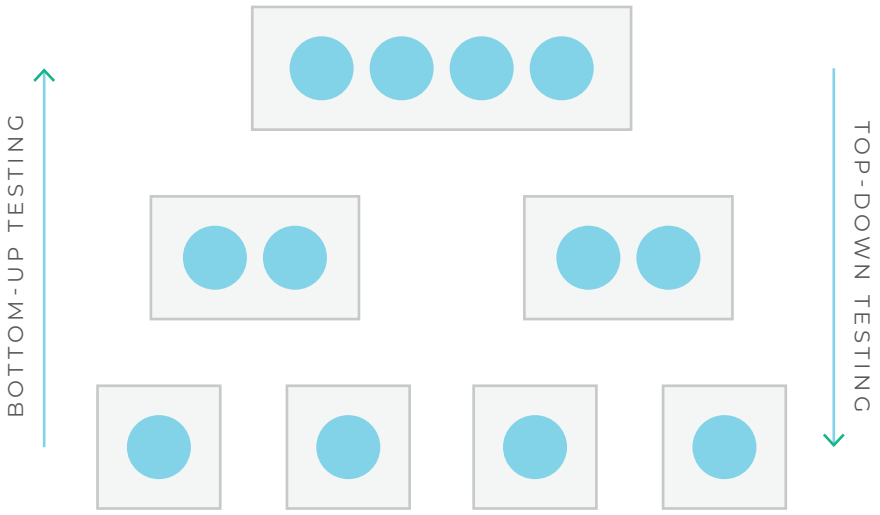
The levels of software testing

- **Component/Unit testing**

The smallest testable part of the software system is often referred to as a unit. Therefore, this testing level is aimed at examining every single unit of a software system in order to make sure that it meets the original requirements and functions as expected. Unit testing is commonly performed early in the development process by the engineers themselves, not the testing team.

- **Integration testing**

The objective of the next testing level is to verify whether the combined units work well together as a group. Integration testing is aimed at detecting the flaws in the interactions between the units within a module. There are two main approaches to this testing: bottom-up and top-down methods. The **bottom-up** integration testing starts with unit tests, successively increasing the complexity of the software modules under test. The **top-down** method takes the opposite approach, focusing on high-level combinations first and examining the simple ones later.



Integration testing approaches

- **System testing**

At this level, a complete software system is tested as a whole. This stage serves to verify the product's compliance with the functional and technical requirements and overall quality standards. System testing should be performed by a highly professional testing team in an environment as close to the real business use scenario as possible.

- **User acceptance testing**

This is the last stage of the testing process, where the product is validated against the end user requirements and for accuracy. This final step helps the team decide if the product is ready to be shipped or not. While small issues should be detected and resolved earlier in the process, this testing level focuses on overall system quality, from content and UI to performance issues. The acceptance stage might be followed by an alpha and beta testing, allowing a small number of actual users to try out the software before it is officially released.

	Unit testing	Integration	System	Acceptance
Why	To ensure code is developed correctly	To make sure the ties between the system components function as required	To ensure the whole system works well when integrated	To ensure customer's and end user expectations are met
Who	Developers / Technical Architects	Developers / Technical Architects	SDET / Manual QA / Business Analyst / Product Owner	Developer / SDET / Manual QA / Product Owner / Product End Users
What	All new code + refactoring of legacy code as well as Javascript unit Testing	New web services, components, controllers, etc.	Scenario Testing, User flows and typical User Journeys, Performance and security testing	Verifying acceptance tests on the stories, verification of features
When	As soon as new code is written	As soon as new components are added	When the product is complete	When the product is ready to be shipped
Where	Local Dev + Continuous Integration (CI, as a part of the build	Local Dev + CI (part of the build)	Staging Environment	CI / Test Environment
How (tools and methods)	Automated, Junit, TestNG, PHPUnit	Automated, Soap UI, Rest Client	Automated (Webdriver) Exploratory Testing	Automated (Cucumber)

The levels of software testing compared

In agile software development, the testing typically represents an iterative process. While the levels generally refer to the complete product, they can also be applied to every added feature. In this case, every small unit of the new functionality is being verified. Then the engineers check the interconnections between these units, the way the feature integrates with the rest of the system and if the new update is ready to be shipped.

6. The Methods of Software Testing

- **Black box testing**

This method gets its name because a QA engineer focuses on the inputs and the expected outputs without knowing how the application works internally and how these inputs are processed. The purpose of this method is to check the functionality of the software making sure that it works correctly and meets user demands. This method can be applied to any testing level, but is used mostly for system and user acceptance testing.

A QA specialist doesn't consider the internal processes of the product while conducting a test

- **White box testing**

Unlike black box testing, this method requires profound knowledge of the code as it entails testing of some structural part of the application. Therefore, generally, the developers directly involved in writing code are responsible for this type of testing. The purpose of white box testing is to enhance security, the flow of inputs/outputs through the application, and to improve design and usability. This method is mainly used at the unit and integration testing levels.

- **Grey box testing**

This method is a combination of the previous two, since it involves testing of both functional and structural parts of the application. Using this method, an experienced tester has partial knowledge of the internal application structure and based on this knowledge can design test cases while still testing from the black-box perspective. This method is mostly applicable to the integration testing level.

- **Ad hoc testing**

This is an informal testing method as it's performed without planning and documentation. Conducting tests informally and randomly without any formal, expected results, the tester improvises the steps and arbitrarily executes them. Though defects found with this method are more difficult to reproduce given the absence of written test cases, this approach helps find important defects quickly, something which cannot be done with formal methods.

7. The Types of Software Testing

Based on the main objective of the process, the testing can be of different types. Here are the most popular testing types according to the ISTQB survey.

Testing type	Object	Method used	Levels of testing
Functional Testing	Testing software functions	Black Box	User acceptance System
Performance Testing	Testing responsiveness and stability of the system performance under a certain load	Black Box	Any level
Use case Testing	Checking that the path used by the user is working as intended	Black Box	User acceptance System Integration
Exploratory Testing	Validating user experience	Ad hoc	User acceptance System
Usability testing	Checking that the system is easy to use	Black Box	User acceptance System
Security testing	Protecting the system	White Box	System

Most popular software testing types described according to their object, method applied and testing levels during which they are used

- **Functional testing**

Winning 83 percent of the respondents' votes, functional testing is the most important testing type. This is to be expected, since without functionality there would be no use of all other non-functional aspects of the system.

In functional testing, the system is tested against the functional requirements by feeding it input and examining the output. This type of testing applies the black box method. Consequently, it gives significance not to the processing itself, but rather, on its results. Functional testing is usually performed within the levels of system and acceptance.
Typically, the process of functional testing comprises the following set of actions:

- Outlines the functions for the software to perform
- Composes the input data depending on function specifications
- Determines the output depending on function specification
- Executes the test case
- Juxtaposes the received and expected outputs

- **Performance Testing**

Performance testing has been selected by 60.7 percent of respondents as the most important non-functional testing type. Performance testing is aimed at investigating the responsiveness and stability of the system performance under a certain load.

Depending on the workload, a system behavior is evaluated by different kinds of performance testing:

- Load Testing — at continuously increasing workload
- Stress Testing — at or beyond the limits of the anticipated workload
- Endurance Testing — at continuous and significant workload
- Spike Testing — at suddenly and substantially increased workload

- **Use case testing**

It's the most widely used testing technique, followed by exploratory testing. Use case describes how a system will respond to a given scenario created by the user. It is user-oriented and focuses on the actions and the actor, not taking into account the system input and output. Keeping the project concepts in mind, developers write use cases and after completing them, the behavior of the system is tested accordingly. Testers, in their turn, use them to create test cases.

Use case testing is applied widely in developing tests at system or acceptance levels. It also helps uncover the defects in integration testing. Use case testing checks whether the path used by the user is working as intended and makes sure the tasks can be accomplished successfully. Applying use case testing, analysts can detect shortcomings and modify the system so that it attains efficiency and accuracy.

- **Exploratory Testing**

The exploratory testing technique was first described by Cem Kaner, a software engineering professor and consumer advocate, as:

> *"a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the value of her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project."*

Using the ad hoc method, exploratory testing does not rely on predefined and documented test cases and test steps as most testing types do. Instead, it is interactive and free-form process, with the main focus on validating user experience, not code. It has much in common with the ad hoc or intuitive testing, but is more systematic. Applying exploratory testing, skilled testers can provide valuable and auditable results.

- **Usability Testing**

Chosen by 44.1 percent of respondents, usability testing is performed from the end-user's perspective to see if the system is easy to use. This testing type is not to be confused with user acceptance testing. The latter verifies that the final product meets the set requirements; the former ensures that the implementation approach will work for the user.

- **Test automation**

To speed up and improve the quality of software testing and improve its quality, it's important to adopt advanced automation.

Test automation is critical in terms of continuous testing as it eases the burden of managing all of the testing needs, allowing more time and effort to be spent on creating effective test cases. The test automation tendency is supported by the ever-growing adoption of agile methodologies, which promote both test automation and continuous integration practices as the cornerstone of effective software development. We invite you to check our article that compares the most popular automated testing tools including Selenium, TestComplete, and Ranorex.

8. Test Automation

To speed up and improve the quality of software testing and improve its quality, it's important to adopt advanced automation.

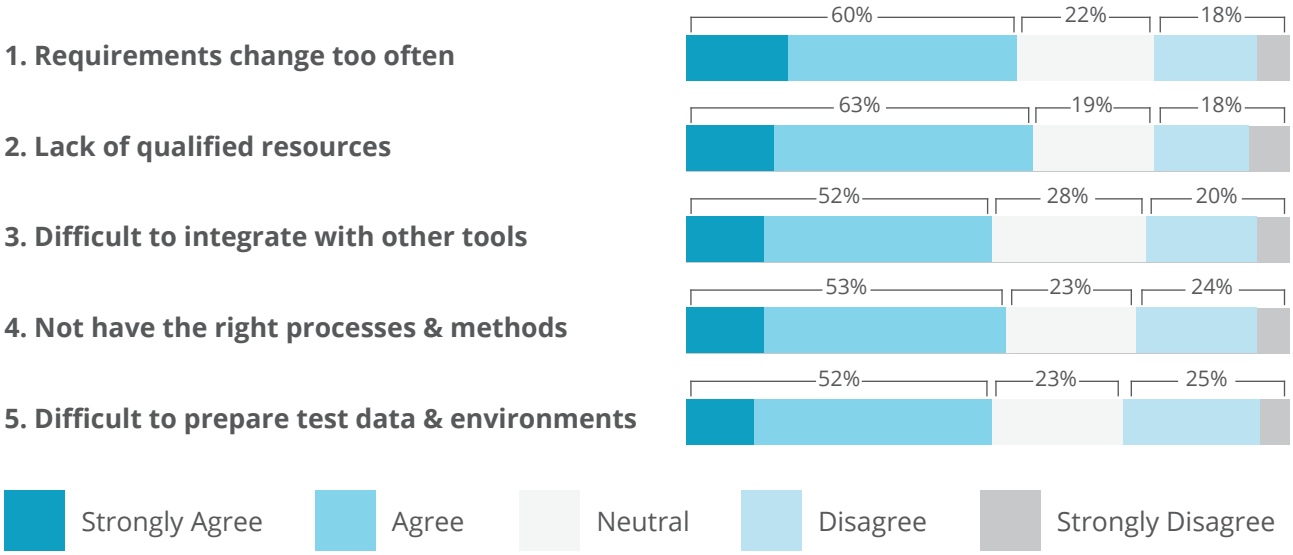
Test automation is critical in terms of continuous testing as it eases the burden of managing all of the testing needs, allowing more time and effort to be spent on creating effective test cases. The test automation tendency is supported by the ever-growing adoption of agile methodologies, which promote both test automation and continuous integration practices as the cornerstone of effective software development. We invite you to check our article that compares the most popular automated testing tools including Selenium, TestComplete, and Ranorex.

The process of test automation is typically conducted in several consecutive steps:

- Preliminary Project Analysis
- Framework Engineering
- Test Cases Development
- Test Cases Implementation
- Iterative Framework Support

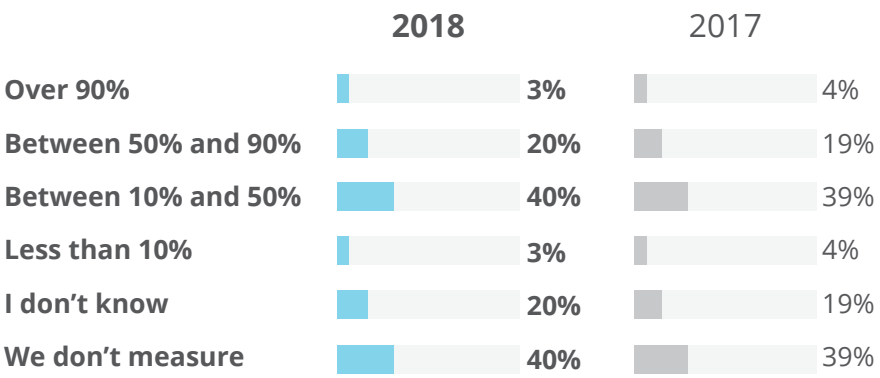
Benefits of test automation. Automation can be applied to almost every testing type, at every level. As a result, the automation minimizes the human effort required to efficiently run tests, reduces time-to-market and the cost of errors because the tests are performed up to 10 times faster, when compared to manual testing process. Moreover, such a testing method is more efficient as the framework covers over 90 percent of the code, unveiling the issues that might not be visible in manual testing, and can be scaled as the product grows.

Test automation in numbers. According to the ISTQB® survey, 64.4 percent of their respondents vote for test automation activities as the main improvement area in software testing. At the same time 43,4 percent of the respondents name test automation the top challenge in Agile projects. Here are the most striking problems faced in applying test automation based on the survey by Katalon Studio.



Challenges in applying test automation

The gradual growth of the contribution of test automation is confirmed by the following survey.



Automation in your Company

Automated test activities include test execution, functional test case design, test data generation, and testing of end-to-end business scenarios.

However, the most effective testing approaches combine manual and automated testing activities in order to achieve the best results.

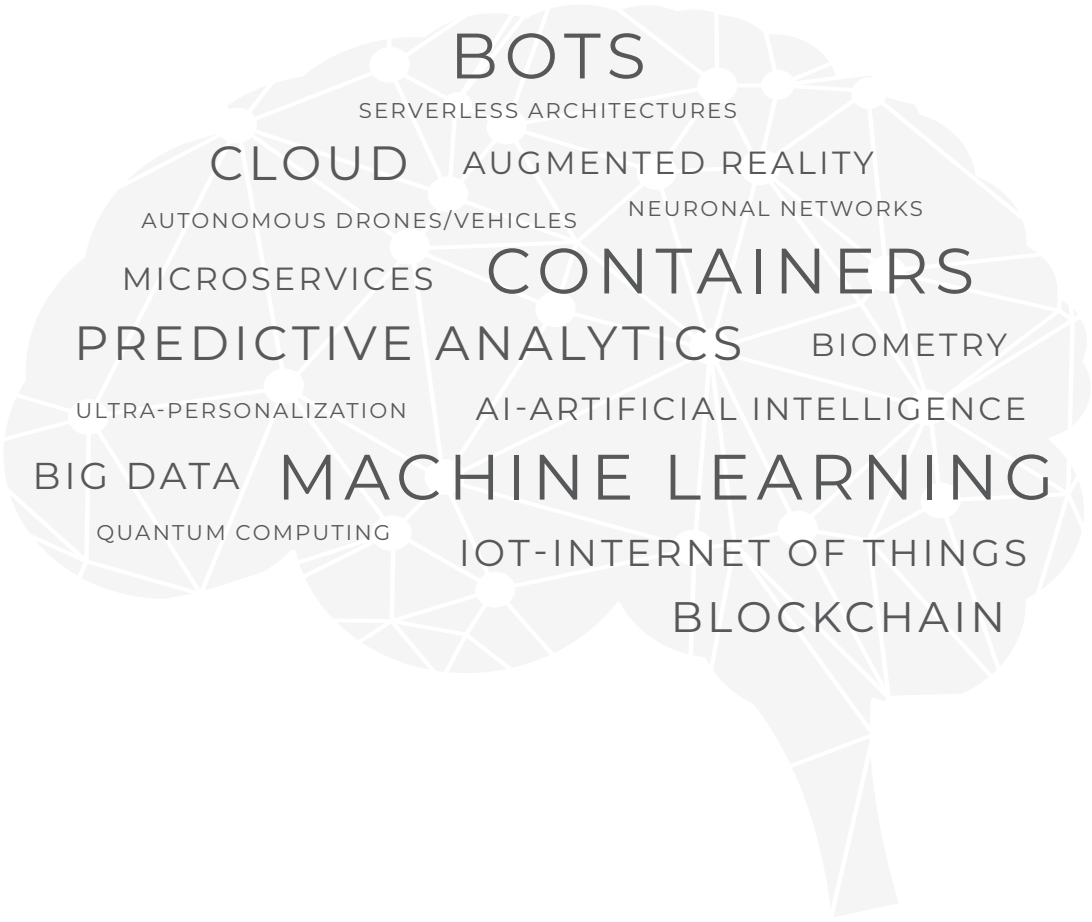
9. Regression testing

Regression testing is the practice of verifying software behavior after updates to ensure that the changes haven't impacted existing system functions, stability, and overall integrity. Regression testing can be applied to all levels and with all types of testing procedures but the most common way is to run regression testing according to use cases. Regression quality assurance workflow can be automated to avoid repetitive manual tests after each update. There are multiple regression testing techniques:

- Retesting all test cases
- Selecting specific test cases
- Prioritizing test cases to verify the most critical ones first and then test the rest
- Hybrid techniques

10. The Future of Testing

As a part of technological progress, testing is continually evolving to meet ever-changing business needs as it adopts new tools that allow the tester to push the boundaries of quality assurance.



"Hot topics" in software testing in the coming years according to the PractiTest survey

New subjects expected to affect software testing in near future are security, artificial intelligence, and big data.

- **Security**

The World Quality Report survey shows that security is one of the most important elements of an IT strategy. Input from security is vital to protecting the business. Security vulnerabilities can seriously tarnish brand reputation. For those reasons, test environments and test data are considered the main challenges in QA testing today.

Data protection and privacy laws also raise concerns about the security of test environments. If an environment contains personal test data and suffers a security breach, businesses must notify the authorities immediately. As a result, it is so important for test environments to be able to detect data breaches.

Most popular in cloud environments, security testing intends to uncover system vulnerabilities and determine how well it can protect itself from unauthorized access, hacking, any code damage, etc. While dealing with the code of application, security testing refers to the white box testing method.

The four main focus areas in security testing:

- Network security
- System software security
- Client-side application security
- Server-side application security

It is highly recommended that security testing is included as part of the standard software development process.

- **Artificial intelligence**

The challenges of testing are increasing and their solutions have unlimited number of situations requiring artificial intelligence to test them thoroughly. Different implementations of AI using machine learning-based algorithms will soon become embedded in applications to perform tasks once reserved for humans.

Although test automation solutions in the intelligence area are not well-established yet, the shift towards more intelligence in testing is inevitable. Cognitive automation, machine learning, self-remediation, and predictive analysis are promising emerging techniques for the future of test automation.

That said, a Boston-based startup mabl already simplifies functional testing by combining it with machine learning. “As we met with hundreds of software teams, we latched on to this idea that developing... is very fast now, but there’s a bottleneck in QA,” says Izzy Azeri, a co-founder of mabl. “Every time you make a change to your product, you have to test this change or build test automation.”

With mabl there is no need to write extensive tests by hand. Instead, you show the application the workflow you want to test and the service performs those tests. Mabl can even automatically adapt to small user interface changes and alert developers to any visual changes, JavaScript errors, broken links, and increased load times.

Adopting smarter automation solutions will be essential for testing the emerging intelligent applications and products in their rapidly changing business environments.

- **Big data**

Currently, the two major concerns regarding test data management are data compliance and big data.

First, with many onerous protection laws arriving on the scene, simply copying real-world data presents a risk of violating them. For example, the EU’s General Data Protection Regulation (GDRP) became law in May 2018 for all companies operating in the EU. Given the threat of significant fines, data compliance concerns are on the front burner of most IT departments today.

Second, managing huge volumes of data that are constantly uploaded on various platforms demands a unique approach for testing as traditional techniques can no longer cope.

Big data testing is aimed at checking the quality of data and verifying data processing. Data quality check involves various characteristics like conformity, accuracy, duplication, consistency, validity, data completeness, etc. Data processing verification comprises performance and functional testing. Big data testing demands a high level of testing skills as the processing is very fast.

Conclusion

In 2012, Knight Capital Americas, a global financial firm, experienced an error in its automated routing system for equity orders - the team deployed untested software to a production environment. As a result, the company lost over $460 million in just 45 minutes, which basically led to its bankruptcy.

History knows many more examples of software incidents which caused similar damage. Yet, testing remains one of the most disputed topics in software development. Many product owners doubt its value as a separate process, putting their businesses and products at stake while trying to save an extra penny.

Despite a widespread misbelief that a tester's only task is to find bugs, testing and QA have a greater impact on the final product success. Having deep understanding of the client's business and the product itself, QA engineers add value to the software and ensure its excellent quality. Moreover, applying their extensive knowledge of the product, testers can bring value to the customer through additional services, like tips, guidelines and product use manuals. This results in reduced cost of ownership and improved business efficiency.

References

1. http://softwaretestingfundamentals.com/software-quality/
2. http://www.davidchappell.com/writing/white_papers/The_Three_Aspects_of_Software_Quality_v1.0-Chappell.pdf
3. https://www.amazon.com/Foundations-Software-Testing-ISTQB-Certification/dp/1844809897
4. https://www.amazon.com/Software-Engineering-Practitioners-Roger-Pressman/dp/0073375977/
5. http://www.satisfice.com/presentations/strategy.pdf
6. http://www.testingexcellence.com/test-strategy-and-test-plan/
7. https://standards.ieee.org/findstds/standard/829-2008.html
8. https://www.istqb.org/documents/ISTQB%202017-18_Revised.pdf
9. http://kaner.com/?p=46
10. https://www.ibm.com/developerworks/community/blogs/Govind_Baliga/entry/defining_exit_criteria_for_testing5?lang=en
11. http://www.ibm.com/developerworks/rational/library/apr06/rose/
12. https://www.sec.gov/litigation/admin/2013/34-70694.pdf
13. https://www.sogeti.com/globalassets/global/downloads/testing/wqr-2017-2018/wqr_2017_v9_secure.pdf
14. https://d1h3p5fzmizjvp.cloudfront.net/wp-content/uploads/2018/06/05101901/The-Most-Striking-Problemns-in-Test-Automation-A-Survey.pdf
15. http://qablog.practitest.com/wp-content/uploads/2018/07/2018_state_of_testing_report_1.2.pdf?full_name=Ksenia&email=annabond1995%40gmail.com&utm_source=qablog&utm_medium=button&utm_campaign=&utm_term=&utm_content=download&page_type=resource%2C&role=Other&industry=Marketing%2C%20advertising%20and%20media&referrer=
16. https://www.softwaretestinghelp.com/devops-and-software-testing/
17. https://qaconsultants.com/wp-content/uploads/2015/10/What-is-the-best-automation-testing-approach-Final.pdf

About AltexSoft

AltexSoft is a Technology & Solution Consulting company co-building technology products to help companies accelerate growth. The AltexSoft team achieves this by leveraging their technical, process and domain expertise and access to the best price-for-value Eastern European engineers. Over 100 US-based and 200 worldwide businesses have chosen the company as their Technology Consulting Partner.

US Sales HQ

701 Palomar Airport Road,

Suit 300, Carlsbad, CA 92011

+1 (877) 777-9097

Global HQ

32 Pushkinskaya Str.,

Kharkiv, Ukraine 61057

+38 (057) 714-1537