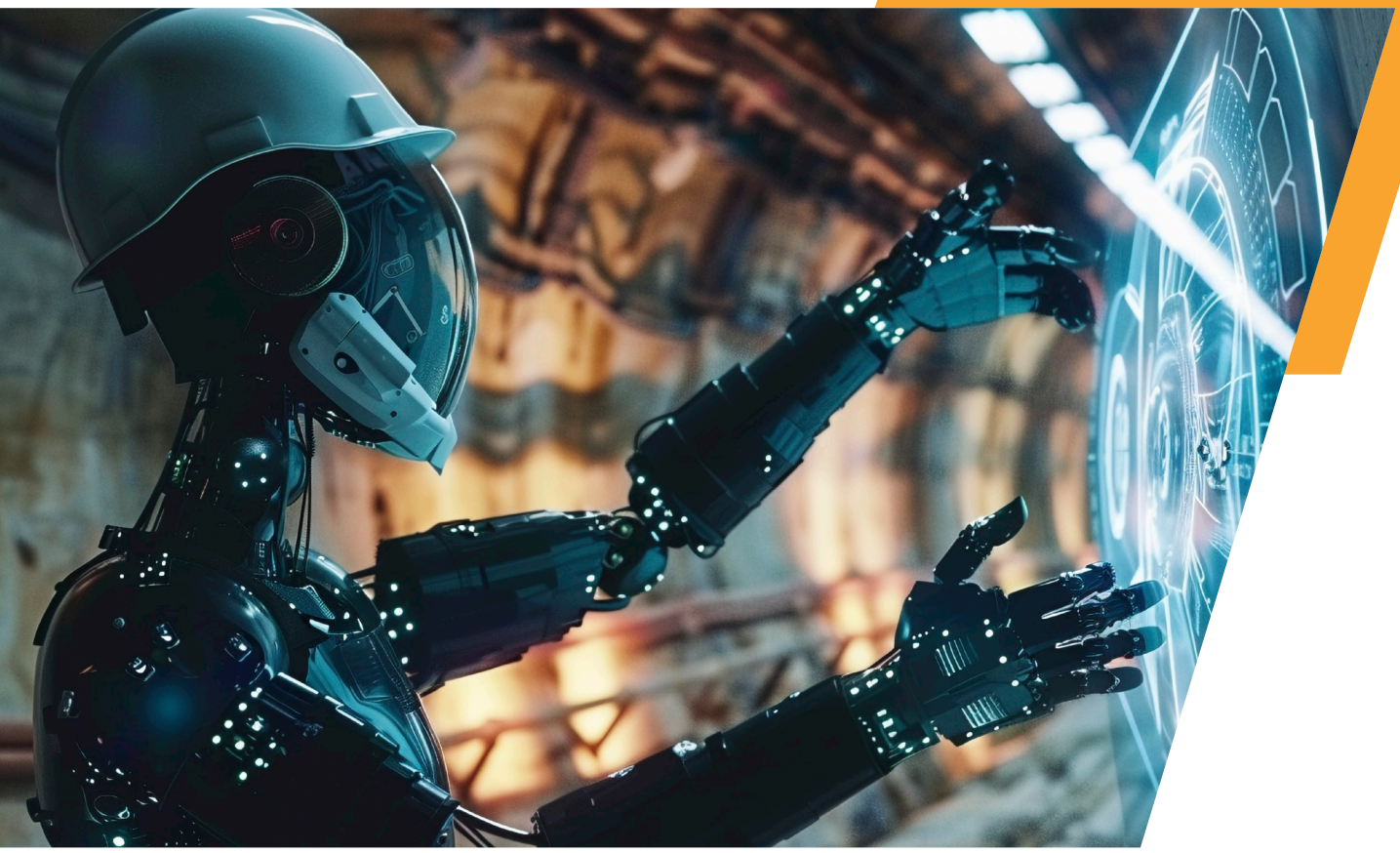
BOTM™

Codeless Testing Automation



OPTIMIZING TEST COVERAGE

INDEX

INTRODUCTION

The software world is a paradoxical one, and balancing the paradoxes can be quite a tightrope walk. On one hand is the non-negotiable demand for 'Superior User Experience' which necessitates incorporating fast paced digital innovations and ensuring they work perfectly – to consistently satiate changing user preferences. On the other hand is the all-important 'Go-to-market Race', which leaves extremely crunched time for coping with the fast changing digital landscape. This enigma, often leads to testing being sacrificed on the altar of time. Unfortunately, it can prove extremely detrimental to the app, as an insufficiently tested app is a ticking time-bomb waiting to explode and decimate the goal of superior user experience.

It is in this scenario that Optimizing Test Coverage gains importance, as it is geared to help QA professionals improve the quality of test cases to ensure there are no gaps in meeting testing goals. Test Coverage which defines what percentage of application code is tested and whether the test cases cover all the codes, thus becomes a vital yardstick to qualitatively gauge the quantum of testing met by a set of tests. It also provides valuable information on adequate coverage of all testing requirements, which becomes the bedrock for launching apps and releasing updates with confidence.

Given the fact that the fast paced digital world is here to stay, this Whitepaper aims at helping QA professionals master the tightrope walk, by providing constructive insights into how to optimize test coverage and meet the twin testing goals of 'Superior User Experience' and 'Faster Go-to-market Time'.

ABSTRACT

Optimizing Test Coverage is an important step in the direction of app success, and consequently a means to promoting the interests of stakeholders, as it seeks to validate the adequacy of testing.

This Whitepaper unravels how test coverage can be optimized and reviews it in two sections:

- **Test Coverage – An Overview**
- **Optimizing Test Coverage while Minimizing Time**

The first section 'Test Coverage – An Overview' presents an understanding of what test coverage is about and covers the following topics:

- **Types of Test Coverage**
- **The Various Nuances of Test Coverage**
- **Calculating Test Coverage**
- **Advantages of Test Coverage for the Project**

The second section 'Optimizing Test Coverage while Minimizing Time' explores the following:

1. **The Test Plan**
    - **Code Coverage**
    - **Data Focused Coverage**
    - **Platform Related Coverage**

2. **Test Coverage Optimization**
    - **Strategies for Effective Test Coverage**
    - **Guidelines for Efficient Test Coverage Optimization**
    - **Techniques for Optimizing Testing Time**

It is hoped that the insights gained will help create well tested apps that are winners in the 'User Satisfaction contest', thus gaining market supremacy – to benefit all stakeholders.

TEST COVERAGE – AN OVERVIEW

The starting point of implementing any good concept is to understand it in all its nuances, and this is true for optimizing test coverage too. Test Coverage is basically an indicator which helps determine how much testing is done and whether it is qualitatively adequate. It is the amount of testing done on a specified set of requirements.

This can be better understood by an example. A smart-watch, which is perfectly tested to guarantee precise display of time; fitness parameters; and mobile phone functions; and also tested for weather conditions and sturdiness; is still not adequately tested until it's strap is tested for durability, flexibility, and comfortable fit on various wrist sizes. A smart-watch with topnotch mechanism but a poorly designed strap is bound to backfire on the customer satisfaction scale. Undoubtedly, test coverage of core functions are extremely critical, yet even the non-core requirements of the product need relevant testing to make it a winner in the market space.

Coming to software testing, it is vital that Test Coverage must include requirements like installation, configuration, performance, speed, security, safety, etc.; and that these are thoroughly tested before the app's go-live. For app success, Test Coverage must cover all that's necessary for users' expectations to be satisfactorily met under varying situations and environments.

To get a better understanding of Test Coverage, it is important to distinguish it from Code Coverage. Code Coverage refers to unit testing processes done by developers to ensure that all areas of the code are covered at least once. Test Coverage, however, is done by the QA team and focuses on testing every covered requirement at least once. It is the team that decides what's important and necessary to be deemed as a covered requirement.

01 Types of Test Coverage

Test Coverage can be broadly classified into two categories viz. Functional Coverage, and Non-functional Coverage.

Functional Coverage

As the name suggests, Functional Coverage quantifies the degree to which tests cover the functional requirements of the software. It aims at confirming that all relevant conditions have been tested to ensure that the software performs as per expectations. Another concept that needs attention here is Branch Coverage, which quantifies the number of branches in the code that have been tested. Smoke Testing, Unit Testing, and Database Testing are all types of Functional Testing.

Non-Functional Coverage

Non-functional coverage measures the degree to which non-functional elements have been tested and indicates the percentage of the type/s of the elements being covered. E.g. For a mobile app, using traceability between tests and supported devices, the rate of devices that are dealt with by Compatibility Testing can be calculated, making it possible to detect coverage gaps. Examples of Non-functional Testing are – Performance Testing, Usability Testing, and Volume Testing which relate to behavioral features of the system.

It can be deduced that non-functional testing is designed to test the app's preparedness, for variables that were not part of the functional testing coverage.

02 The Various Nuances of Test Coverage

Test Coverage can be broadly classified into two categories viz. Functional Coverage, and Non-functional Coverage.

Product Coverage

Product Coverage will help determine the areas of the product that have been tested and those that are not tested. As seen earlier, in the smart-watch example, there are various core and non-core, technical and non-technical aspects that need to be tested. Product Coverage will provide valuable information about the extent and adequacy of product testing done. When testing apps, while the app's overall functioning is a given for product testing, it's also important to test how the app functions when used in conjunction with other apps; to ensure that an uncommon activity doesn't trigger app failure; that relevant error and alert messages provide timely warning to users; that the app is easy to maneuver and understand; that the help menu is easily accessible, useful, simplified and clear. These are areas that may tend to be overlooked when time is of essence, but this can prove detrimental to the app and to the goal of superior user experience.

Risk Coverage

Risk Coverage is another important component of Test Coverage as it measures gaps in the testing of the safety and security aspects of the app, which are vital given that apps contain sensitive user information. For Mobile Apps, some of the major security risks that need QA attention are: inadequate API protection; vulnerable server controls; insufficient load testing; client-side injections; risky sensitive data storage protocols; hardcoded password or keys; poor source code security;

leakage of confidential data; insecure data transmission; inadequate logging and monitoring controls. A Banking App for example has very sensitive data and hence Risk Coverage is vital since data leakage protection is absolutely non-negotiable. But there are also load testing risks that need to be taken care of. A case in point is the demonetization period that saw a flurry of banking activity, which needed Banking apps to support these unprecedented loads. It's the same with e-commerce platforms that need to provide for unusual digital traffic when discounts and special sales are announced.

Requirements Coverage

Requirement Coverage basically looks at which of the decided requirements have been tested. This is important as the requirements as defined in the requirement document, have been arrived at considering customer's requirements and preferences. Delivering fancy features that don't cater to customers' requirements will adversely affect app demand and usage. Furthermore, the requirement document will form the basis for other departments to take forward their jobs too. The marketing department for example, will design their ads and target groups depending on the features the app is supposed to cater to. But if a developer incorporates topnotch unlisted features, and testers test all of these, but overlook testing of even one specified requirement, it will have uncomfortable repercussions when customers seeking the untested feature buy the app, only to feel cheated when they find it absent or ill-functioning. Hence requirement coverage is a very important part of Test Coverage and can go a long way in enhancing user satisfaction.

03 Calculating Test Coverage

To evaluate Test Coverage, the following formula can be used:

Test Coverage = No. of Test Cases Executed * 100/Total No. of Test Cases

However, it's also important to know when to calculate coverage, because doing it too early in the process, will result in many gaps, as things are incomplete. Ideally the calculation should be done after the Last Build i.e. Final Regression Build, to get a correct coverage of the tests performed for the given requirements.

For proactive implementation of any concept, it's important to understand how useful and beneficial it is. Hence, here's a peek into how Test Coverage benefits the project as well as QA personnel.

Advantages of Test Coverage for the Project

- Helps prioritize testing tasks by distinguishing between critical and non-critical test cases
- Helps attain 100% requirement coverage and prevents requirement leakage
- Promotes traceability between generated test cases and defined requirements
- Facilitates Impact Analysis
- Useful in tracking all build cycles and fixes
- Helps differentiate between releases, and refines requirements
- Assists in fulfilling functionality coverage as per client's needs
- Greatly helps to define the EXIT criteria
- Augments precision in preparation of Test Closure Report

Clearly Test Coverage proves to be helpful to multiple stakeholders including the QA team and the client, and hence the next section of this Whitepaper will delve into ways in which Test Coverage can be optimized and testing time can be rationalized.

OPTIMIZING TEST COVERAGE WHILE MINIMIZING TIME

Test Coverage Optimization is a product of insightful planning, coupled with resource and time optimization. Hence the first step is to review the carefully prepared 'requirement document', define timelines, and take stock of resource availability – both man and machine. The deadline for app release becomes the sacrosanct target, and interim timelines must be set for meeting the various milestones in the testing process. This will avoid delay in the app's release date, and help clinch the 'go-to-market' race, which impacts app success.

It's imperative to ensure that all tasks in the requirement document are assigned to the available QA professionals, before the testing process begins, because a single mandatory task left out, has the potential to bring to naught the diligent collective effort of the entire team. In order to reap learning curve benefits, and achieve more in less time, at lower costs, it greatly pays to assign tasks according to the skills of testers, assigning complex tasks to more experienced testers, and leaving the simpler tests to average testers.

The Test Plan is an important part of optimizing test coverage and includes the following critical areas:

1. **Code Coverage**
2. **Data Focused Coverage**
3. **Platform Related coverage**

1. Code Coverage

The various parameters for ensuring optimal Code Coverage are tabled below for easy reference.

S. No.	Coverage Type	Particulars
1.	Functional Coverage	Verify all the functions and database store procedures related to the functionalities.
2.	Statement Coverage	Confirm each line of code.
3.	Condition Coverage	Verify all the loops and conditions in the code.
4.	Path Coverage	Detect the potential paths from the starting point in the code which has been executed.
5.	Entry and Exit Coverage	Confirm how many functions/procedures were executed from start to finish.

These parameters are inter-linked to various degrees. While the Path Coverage is inter-linked with Condition, Statement and Entry/Exit Coverage, the Statement Coverage is independent of Condition Coverage.

2. Data Oriented Coverage

Data Oriented Coverage checks if combinations of data values have occurred. It also seeks to test some of the data using all possible values at least once. The domain defines which input and output parameters are to be considered. Data Oriented Coverage can be achieved by writing coverage groups, coverage points, and via cross coverage. It is applicable for white box as well as black box testing. Data oriented testing includes the following:

- Equivalence Partitioning
- Boundary Value Analysis
- Data Combination Testing (e.g. using pair-wise or n-wise testing)
- Data Cycle Testing (using CRUD)
- Data Flow Testing

3. Platform-related coverage

Platform-related coverage is vital to ensure test coverage for all platforms on which the app will be used i.e. Web, Desktop, and Mobile. The high level of Mobile fragmentation compounds coverage complexities as there are a substantial number of different OS versions available and operational in the digital world. It is therefore necessary to pay detailed attention to this platform coverage to ensure superior user experience across Mobiles and OS versions. Platform-related Coverage must also ensure test coverage for various browsers.

Having reviewed test coverage in its diversity, it's important to understand how to optimize test coverage, within the limited available time and resources. The remaining part of this Whitepaper will unfold this aspect, exploring it from three different angles.

01 Test Coverage Optimization

1. Strategies for Effective Test Coverage
2. Guidelines for Efficient Test Coverage Optimization
3. Techniques for Optimizing Testing Time

1. STRATEGIES FOR EFFECTIVE TEST COVERAGE

Specification-based Test Coverage (Black Box Testing)

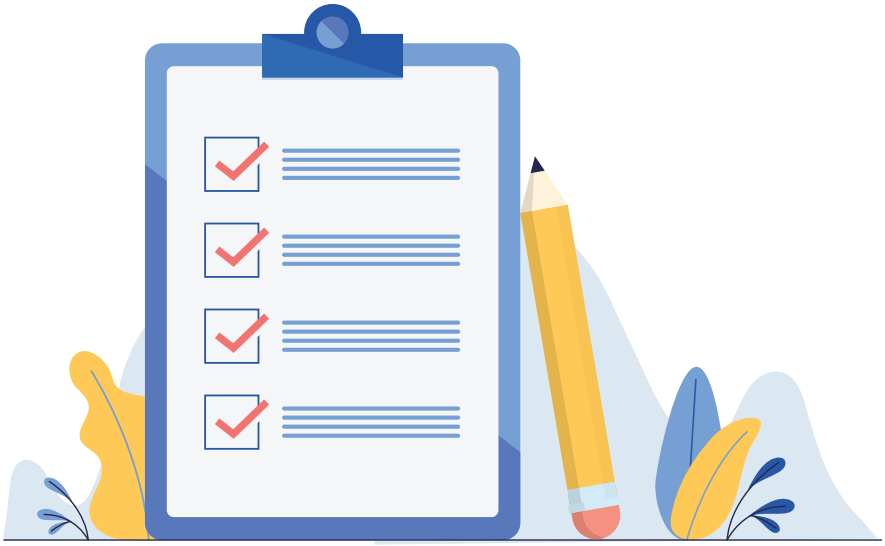
This strategy evaluates the app based on its requirements and specifications which are used to derive test cases, without considering its internal structure or implementation details. The specifications can be functional or non-functional and can be at different levels of abstraction i.e. user requirements, system requirements, or design specifications. The focus is to write test cases that cover as many scenarios described in the specifications, so that the software satisfies the intended criteria; functions as intended; and meets user expectations.

Structure-based Test Coverage (White Box Testing)

Structure-based test coverage is a code-based testing strategy where the tester has knowledge of the internal structure of the system. Test cases are written after analyzing the internal structure based on code, branch, path, and condition coverage. It can be achieved with the help of Branch Testing, Statement Testing, etc.

Experience-based Test Coverage

As the name suggests, the experience-based strategy relies on the tester's experience with testing, development, similar applications, the same application in previous releases, and the domain itself. Testers' experiential knowledge forms the basis for designing the test cases.



2. GUIDELINES FOR EFFICIENT TEST COVERAGE OPTIMIZATION

- **Resource Shuffling**

  Swapping tasks between testing teams helps to weed out any escaped bugs; and also promotes knowledge sharing as well as team members' involvement in the project.

- **Compatibility Coverage**

  Cross browser and cross platform testing is imperative to generate confidence in the app's compatibility with diverse user preferences.

- **Ownership**

  Giving testers ownership for the modules assigned to them brings creativity, accountability and increased responsibility to the testing process.

- **Deadlines**

  The app's release date should be clearly set and this should be treated by all teams as inviolable; so that in turn they can plan their own interim timelines and work to achieving the targeted release date.

- **Communication**

  Effective communication between QA teams, and also between all stakeholders like the development team, the app owners/client etc. is vital for optimizing test coverage, as it ensures that everyone is on the same page throughout the SDLC, and that test coverage is proceeding in the right direction.

- **Requirements Traceability Matrix (RTM)**

  RTM helps confirm that all requirements outlined for a system are linked at every point during the verification process, and also ensures that they are duly tested as per test parameters and protocols. This greatly helps stakeholders take informed decisions for the release schedule.

- **Collaborative Tools**

  The right testing tool can greatly contribute to optimizing test coverage, as its speed and accuracy optimize time, as well as improves quality; widens test coverage; and greatly increases efficiency of the testing process.

3. TECHNIQUES FOR OPTIMIZING TESTING TIME

- It greatly pays to be familiar with the functionalities for the decided requirements and specifications, in order to be able to evaluate and prioritize the tasks correctly
- Choosing the right test automation tool is also important. It's imperative to note that this is not necessarily the best available tool in the market, but must be the one most suitable to the project
- Prioritizing requirements into critical, major, and minor categories helps to optimally channelize testing time and effort
- Impact Analysis which assesses the impact of earlier releases can offer valuable time-saving insights
- Being aware of how a release is different from the previous one, can help identify critical requirements more accurately and focus on maximum positive coverage
- A checklist for all the basic interactions helps in including efficient tasks; and also ensures that pending tasks are completed
- Build Management is a good way to boost testing efficiency, as it helps stakeholders including product owners, to keep track of all fixes, versions, and impact on the current releases
- Creating test data directly from the database can bring ease and time-efficiency by reducing UI interactions, and increasing testing speed and reliability
- Another useful tip is to execute the most important business test cases on all browsers every time; and run the low priority test cases on a single browser, using a different browser for each test suite
- It is hoped that these strategies, guidelines and techniques will help the reader successfully optimize test coverage, achieving more in less time; to confidently release the app right on schedule

CONCLUSION

Optimizing Test Coverage is an important strategy for app success, as it focuses on efficient and relevant testing, within the constrained time and monetary budgets. These constraints mandate that testing should be smartly done, emphasizing the qualitative rather than merely quantitative aspects. It is a proven fact, that more testing does not necessarily mean better testing. What's important is to have a well-thought out strategy which widens 'requirement-based test coverage' within the given constraints, without compromising on testing quality. This is the crux of what Optimizing Test Coverage is all about. A structured approach, which targets '100% requirement coverage' using effective testing methods and tools, will help achieve qualitative testing despite the prevailing constraints. Using the right testing tool is a must for effectively optimizing test coverage in minimal time.

For Mobile App Testing, BOTm offers features that redefine industry standards. Being a fully automated testing platform that consistently incorporates the latest in technological advancements, BOTm is a proven tool that optimizes test coverage.

Visit **www.botmtesting.com** and sign up for a **Free Trial**, to explore the spot-on features and manifold benefits that BOTm offers.

GET IN TOUCH

022 4050 8200

sales@botmtesting.com | www.botmtesting.com

BOTm is the accelerator BOT for automated and manual testing of mobile applications - developed for both Android and iOS devices.