**Software Testing Metrics, its Types and Example**

Software testing metrics are quantifiable indicators of the [software testing](https://www.geeksforgeeks.org/software-testing-basics/) process progress, quality, productivity, and overall health. The purpose of software testing metrics is to increase the efficiency and effectiveness of the software testing process while also assisting in making better decisions for future testing by providing accurate data about the testing process. A metric expresses the degree to which a system, system component, or process possesses a certain attribute in numerical terms. A weekly mileage of an automobile compared to its ideal mileage specified by the manufacturer is an excellent illustration of metrics. Here, we discuss the following points:

## **Importance of Metrics in Software Testing:**

* **Early Problem Identification**
* **Allocation of Resources**
* **Monitoring Progress**
* **Continuous Improvement**

## **Types of Software Testing Metrics:**

* **Process Metrics**
* **Product Metrics**
* **Project Metrics**

## **Manual Test Metrics: What Are They and How Do They Work?**

Manual testing is carried out in a step-by-step manner by quality assurance experts. Test automation frameworks, tools, and software are used to execute tests in automated testing. There are advantages and disadvantages to both human and automated testing. Manual testing is a time-consuming technique, but it allows testers to deal with more complicated circumstances. There are two sorts of manual test metrics:

**1. Base Metrics:**Analysts collect data throughout the development and execution of test cases to provide base metrics. By generating a project status report, these metrics are sent to test leads and project managers. It is quantified using calculated metrics.

* The total number of test cases
* The total number of test cases completed.

**2. Calculated Metrics:**Data from base metrics are used to create calculated metrics. The test lead collects this information and transforms it into more useful information for tracking project progress at the module, tester, and other levels. It’s an important aspect of the SDLC since it allows developers to make critical software changes.

## Other Important Metrics:

1. **Defect metrics**
2. **Schedule Adherence**
3. **Defect Severity**
4. **Test case efficiency**
5. **Defects finding rate**
6. **Test Coverage**
7. **Defect Fixing Time**
8. **Defect cause**

## **Test Metrics Life Cycle:**

1. **Analysis:**
   * The metrics must be recognized.
   * Define the QA metrics that have been identified.
2. **Communicate:**
   * Stakeholders and the testing team should be informed about the requirement for metrics.
   * Educate the testing team on the data points that must be collected in order to process the metrics.
3. **Evaluation:**
   * Data should be captured and verified.
   * Using the data collected to calculate the value of the metrics
4. **Report:**
   * Create a strong conclusion for the paper.
   * Distribute the report to the appropriate stakeholder and representatives.
   * Gather input from stakeholder representatives.

**Test Case Effectiveness:**

**Test Case Effectiveness = (Number of defects detected / Number of test cases run) x 100**

**2. Passed Test Cases Percentage:**Test Cases that Passed Coverage is a metric that indicates the percentage of test cases that pass.

**Passed Test Cases Percentage = (Total number of tests ran / Total number of  tests executed) x 100**

**3. Failed Test Cases Percentage:**This metric measures the proportion of all failed test cases.

**Failed Test Cases Percentage = (Total number of failed test cases / Total number of  tests executed) x 100**

**4. Blocked Test Cases Percentage:**During the software testing process, this parameter determines the percentage of test cases that are blocked.

**Blocked Test Cases Percentage = (Total number of blocked tests / Total number of  tests executed) x 100**

**5. Fixed Defects Percentage:**Using this measure, the team may determine the percentage of defects that have been fixed.

**Fixed Defects Percentage = (Total number of flaws fixed / Number of defects reported) x 100**

**6. Rework Effort Ratio:**This measure helps to determine the rework effort ratio.

**Rework Effort Ratio = (Actual rework efforts spent in that phase/ Total actual efforts spent in that phase) x 100**

**7. Accepted Defects Percentage:**This measures the percentage of defects that are accepted out of the total accepted defects.

**Accepted Defects Percentage = (Defects Accepted as Valid by Dev Team / Total Defects Reported) x 100**

**8. Defects Deferred Percentage: This measures the percentage of the defects that are deferred for future release.**

**Defects Deferred Percentage = (Defects deferred for future releases / Total Defects Reported) x 100**

# Software Testing – Estimation Techniques

**Last Updated :**03 Aug, 2022

Test estimation is an important part of a project since test estimation plays a key role in test planning, scheduling project, and allocation of necessary resources. The following topics will be discussed here:

1. **What is Software Test Estimation?**
2. **Why Test Estimation?**
3. **What To Estimate?**
4. **How To Estimate?**
5. **Test Estimation Best Practices**
6. **Other Techniques**

Let’s start discussing each of these topics in detail.

### What is Software Test Estimation?

Test estimation comprises efforts contributed by Test Managers or Test Engineers for the completion of testing. Software test estimation starts with the breakdown of work using the fundamental process that includes the identification of various stages, activities, and tasks required to be performed. The set of tasks to be performed are as follows:

* Planning of the tests and controlling their execution.
* Design of the scenarios and Analysis.
* Implementation of the tests and execution.
* Evaluation of exit criteria.
* Test Closure and final sign-off.

Identification of activities for each phase is done and then the tasks and subtasks are defined. After that, we estimate how much time these tasks and subtasks will take to be completed.

### Why Test Estimation?

* Test estimation is done to answer two critical questions, overall cost and time taken. This is an essential aspect for small projects that have a limited budget.
* Test estimation basically acts as a forecast that helps maintain budget and time constraints.
* Estimation is used to predict the efforts required and the time and cost that would be involved in completing the estimated tasks. As soon as the QA team is able to estimate the problem scenario, they can drive the situation effectively and reduce the overall time and budget constraints.
* Test estimation involves experienced people in its core decision-making, they are responsible for implementing best QA practices like test case points, use case point methods, etc.

### What to Estimate?

* **Time:** Time is one of the most critical aspects when it comes to projects as we have deadlines to meet and have SLAs in place for the delivery of projects on time.
* **Resources:**Resources (people, equipment, facilities) are required for carrying out project activities. Proper utilization of resources plays a key part in optimizing available resources and estimating the timelines.
* **Cost:**Cost refers to the project budget where resources and how much cost allocation is required for all the projects are defined.
* **Skillset:** Skillset refers to the knowledge and the experience available among the Team members. A person having a better skillset will take less time to cover the task compared to a person having a lower skillset. This is a crucial aspect when it comes to estimation
* **Risks**involved in the software and the project.

### How To Estimate?

There are 4 stages to do software test estimation:

**1. Divide the whole project into small tasks:**In this step Work breakdown structure (WBS) technique can be used to divide a complete project into modules. The modules are divided into sub-modules. Each sub-modules is divided into functionality, thus the whole project is divided into the smallest tasks.

**Work Breakdown Structure (WBS):** This technique is used when the project is very large and has complex scenarios. Generally, this technique is often used when the testing team is not very sure about the test estimates. In this structure, we divide the project into simpler, and independent tasks. This is a top-down approach.

The approach of WBS:

* Developing a goal.
* Breaking the goal into its deliverables.
* Breaking the deliverables into activities.
* Further breaking the work into small and manageable components thereby allowing more time for resources.

**2. Allocate each task to team members:**In this step, each task is assigned to the appropriate team member in the team.

**3. Estimate the effort required to complete each task:**Two techniques can be used to estimate the effort for tasks. Below are the two software test estimation techniques, Three-point estimation and Function point analysis.

**Three Point Estimation:**Three-point estimation is a technique that makes use of three types of costs or their duration estimates. They are as follows:

* **Optimistic (O):** Best case estimation is estimated with the assumption that the effort is getting channeled correctly and nothing goes wrong. All the scenarios are positive.
* **Most Likely (M):** Most likely estimation is estimated considering most of the things are going well but a problem could resurface in the future. Estimating is considered both positive and negative for specific scenarios.
* **Worst case (W):** Worst case estimation is estimated considering everything goes wrong. Estimation is considered negative for all scenarios.

This technique is used for improving the accuracy of the estimates of cost or duration involved. Here, each task is broken down into minor sub-tasks, and then the results are estimated.

The formula to calculate estimation is:

***Test estimation = (O + (4 \* M) + W) / 6***

***Here,***

* ***O:****Estimation for best case scenarios.*
* ***M:****Estimation for most likely case scenarios.*
* ***W:****Estimation for worst-case scenarios.*

The final result is calculated once we estimate all the case scenarios.

**Example:**Let us assume there is a requirement for testing the entire software application. The application has a lot of parts – Frontend, Backend, Databases, APIs, and performance testing. Now, estimating the possible scenarios:

* **Scenario 1: Team ‘A’** requires **30 man-hours** to test the entire application in the **best-case scenario(O)** i.e. a scenario where no problem arises, everything works as expected and the resources in the team were on the same page in meeting the deadlines and no backlogs were present in the sprint sessions of the SDLC phase.
* **Scenario 2: Team ‘A**‘ requires **45 man-hours**to test the entire application in the **most-likely scenario (M)**i.e. a normal scenario where problems are expected to arise and most of the tasks would be completed with fewer backlogs in the upcoming sprints.
* **Scenario 3**: **Team ‘A**‘ requires **60 man-hours** to test the entire application in the **worst-case scenario (W)** i.e. a scenario where almost the entire situation goes wrong due to a lack of skilled resources or due to the fact that many of the members are newly deployed to a team.

We have three values now:

***O =****30****M =****45****W =****60*

*The average value for the test estimation (E) can be calculated using the formula:*

*E =  (O + (4 \* M) + W) / 6  
   = (30 + 4 \* 45 + 60) / 6  
   = 45 man hours*

***Standard deviation (SD) =****(W – O) / 6  
                                          = (60 – 30) / 6  
                                          = 5*

Hence, the final estimate would be: Team ‘A’ needs 45 +/- 5 person-hours to complete the testing of the application.

**Function Point Estimation:** Function Point Estimation begins with identifying the functions in the software. Once the software function to be tested is identified, it is distributed among the resources available, and the time required to test each function is calculated.

In this estimation technique, we use weights and hence calculate the estimation for entire functions that are under test by multiplying the (weight \* duration). To do this, we do the following steps:

* Identify the smallest function under test and assign the weight as One.
* Finding the time required to test the function.
* Identify other functions as well and similarly calculate the time to test the functions by multiplying the (weight \* duration).
* Once all functions are estimated, we add them to calculate the total effort required for conducting the testing.

This technique introduces us to the concept of **functional point (FP)**. The FP is assigned for the modules having varying complexities – Easy, medium, and hard. The tasks which are simpler as assigned lesser points and for difficult tasks higher number of points are assigned.

**Example:**Let us then take the example of the same application that we took under consideration for Three-Point Estimation. The application has various components: Frontend, Backend, database, and business logic layer. Now, all of the components in the application have easy, medium, and difficult modules. Therefore, segregating the project into modules:

Assume FP = $10 per points.

| **Type of module** | **No: of modules** | **FP (for each module)** | **Total FP (for each module)** |
| --- | --- | --- | --- |
| Easy | 4 | 2 | 8 |
| Medium | 8 | 3 | 24 |
| Difficult | 3 | 5 | 15 |

*Total FP for project = 4 \* 2+ 8 \* 3 +3 \* 5   
                              = 47*

*Therefore, to complete the project we need 470$*

**4. Validate the estimation:**Forward the estimation to the management board, who will review and approve it.

### Test Estimation Best Practices

1. **Resource planning in estimation:**Resource planning and allocation play a key part in estimating the delivery of the project and meeting deadlines.  The availability of resources will not only deliver updates to key stakeholders but also assure that we set our estimations right.
2. **Taking references from past projects:** Some projects may resemble each other with respect to their domain testing or open-source frameworks or the same client. Any difficulties faced in the past should be noted and care must be taken so that they do not recur in the future.
3. **Keeping some buffer time:**Unforeseen circumstances can happen which include the departure of critical team members or unplanned long leaves. This could delay the scheduled timeline of the project and care must be taken to preserve some buffer time.
4. **Sticking to the Estimation:**Whatever estimation we have made, we should try to adhere to that irrespective of its correctness or feasibility. Unless any major change is encountered, we should try not to modify the estimate and stick to it.
5. **Considering the Bug Cycle:** The test estimation has the bug cycle included in it. The encountering of bugs causes a delay in the actual test cycle and hence the number of estimated days increases.
6. **Scope of Project:** Knowing our project objective is important as it helps to estimate between small and large projects and our project delivery depends on it.  Large projects have lots of test scripts, test data, and test documents. Therefore, knowing our objective of testing and planning of deliverables plays a key role in knowing the project scope.
7. **Load Testing Plan:** We need to know the estimated time if there is a requirement to perform Load Testing and hence plan the estimates accordingly.
8. **Parallel Testing:** The scope of parallel testing needs to be known i.e. whether we have the previous versions of the same product for comparing the output. If there is a chance, the testing process becomes easier and product estimation can be done in a better way.

### Other Techniques:

Some other techniques used are:

**1. Wideband Delphi Technique:** This is a technique where experienced people participate and estimate reliably. Here, all the assumptions are taken into consideration, thoroughly discussed, and agreed upon.

**Advantages:**

* This technique is particularly useful when dealing with estimating an effort for a task.
* The technique is completely anonymous and therefore everyone expresses their opinion.
* The technique is relatively simple.
* All the assumptions are equally discussed as well as documented.

**Disadvantages:**

* Management may not agree with the estimation every time.

**2. Use-Case Point Method:**Use-Case Points (UCP) method is an estimation technique used for measuring software with different use cases. This is a very simple technique as there is no additional analysis required.

**Advantages:**

* They are based on use cases and hence can be measured earlier in the Software project lifecycle.
* This technique is independent of skill, size, and experience.
* Estimations are close to actual estimations as they are implemented by experienced team members.

**Disadvantages:**

* The use cases of the UCP techniques are not uniformly structured.
* There is a high impact of Technical and environmental factors on the UCP method.
* More suited when the requirements are in the form of use cases and fail when the method is applied for assigning work within the Team.

**3. Experience-Based Method:**In this method, estimation is banked on the experience of similar projects executed by previous QA teams. This method assumes that the QA team has already tested a project of similar characteristics and uses whether estimating with this technique is feasible or not.

**Advantages:**

* High precision is achieved using this estimation technique.
* Faster Expert based estimation.

**Disadvantages:**

* This technique is not suitable for teams with less skilled resources.
* For unique projects, this technique is difficult to apply.

## **What is Test Plan**:

A [test plan](https://www.geeksforgeeks.org/software-testing-test-plan-estimates-and-strategy/) is a document that consists of all future testing-related activities. It is prepared at the project level and in general, it defines work products to be tested, how they will be tested, and test type distribution among the testers. Before starting testing there will be a test manager who will be preparing a test plan. In any company whenever a new project is taken up before the tester is involved in the testing the test manager of the team would prepare a test Plan.

* The test plan serves as the blueprint that changes according to the progressions in the project and stays current at all times.
* It serves as a base for conducting testing activities and coordinating activities among a QA team.
* It is shared with Business Analysts, Project Managers, and anyone associated with the project.

| **Factors** | **Roles** |
| --- | --- |
| **Who writes Test Plans?** | Test lead, Test Manager, Test Engineer |
| **Who reviews the Test Plan?** | Test Lead, Test Manager, Test Engineer, Customer, Development Team |
| **Who approves the Test Plan?** | Customer, Test Manager |
| **Who writes Test Cases?** | Test Lead, Test Engineer |
| **Who reviews Test Cases?** | Test Engineer, Test Lead, Customer, Development Team |
| **Who approves Test Cases?** | Test Manager, Test Lead, Customer |

## **Why are Test Plans Important**:

The following are some of the key benefits of making a test plan:

* **Quick guide for the testing process:** The test plan serves as a quick guide for the testing process as it offers a clear guide for QA engineers to conduct testing activities.
* **Helps to avoid out-of-scope functionalities:**The test plan offers detailed aspects such as test scope, test estimation, strategy, etc.
* **Helps to determine the time, cost, and effort:**The Test serves as the blueprint to conduct testing activities thus it helps to deduce an estimate of time, cost, and effort for the testing activities.
* **Provide a schedule for testing activities:**A test plan is like a rule book that needs to be followed, it thus helps to schedule activities that can be followed by all the team members.
* **Test plan can be reused:**The test plan documents important aspects like test estimation, test scope, and test strategy which are reviewed by the Management Team and thus can be reused for other projects.

## Objectives of the Test Plan:

1. **Overview of testing activities:** The test plan provides an overview of the testing activities and where to start and stop the work.
2. **Provides timeline:**The test plan helps to create the timeline for the testing activities based on the number of hours and the workers needed.
3. **Helps to estimate resources:** The test plan helps to create an estimate of the number of resources needed to finish the work.
4. **Serves as a blueprint:**The test plan serves as a blueprint for all the testing activities, it has every detail from beginning to end.
5. **Helps to identify solutions:**A test plan helps the team members They consider the project’s challenges and identify the solutions.
6. **Serves as a rulebook:**The test plan serves as a rulebook for following rules when the project is completed phase by phase.

## **Types of Test Plans**:

The following are the three types of test plans:

* **Master Test Plan:**In this type of test plan, includes multiple test strategies and has multiple levels of testing. It goes into great depth on the planning and management of testing at the various test levels and thus provides a bird’s eye view of the important decisions made, tactics used, etc. It includes a list of tests that must be executed, test coverage, the connection between various test levels, etc.
* **Phase Test Plan:**In this type of test plan, emphasis is on any one phase of testing. It includes further information on the levels listed in the master testing plan. Information like testing schedules, benchmarks, activities, templates, and other information that is not included in the master test plan is included in the phase test plan.
* **Specific Test Plan:**This type of test plan, is designed for specific types of testing especially non-functional testing for example plans for conducting performance tests or security tests.

## **Components and Attributes of Test Plan**:

There is no hard and fast rule for preparing a test plan but it has some **standard 15 attributes** that companies follow:



*Components and Attributes of Test Plan*

**1. Objective:** It describes the aim of the test plan, whatever the good process and procedure they are going to follow to give quality software to customers. The overall objective of the test is to find as many defects as possible and to make software bug-free. The test objective must be broken into components and sub-components. In every component following activities should be performed.

* List all the functionality and performance to be tested.
* Make goals and targets based on the application feature.

**2. Scope:**It consists of information that needs to be tested concerning an application. The scope can be divided into two parts:

* **In-Scope:** The modules that are to be tested rigorously.
* **Out Scope:** The modules that are not to be tested rigorously.

**Example:**In an application A, B, C, and D features have to be developed, but the B feature has already been designed by other companies. So the development team will purchase B from that company and perform only integrated testing with A, B, and C.

**3. Testing Methodology:**The methods that are going to be used for testing depend on application to application. The testing methodology is decided based on the feature and application requirements.

Since the testing terms are not standard, one should define what kind of testing will be used in the testing methodology. So that everyone can understand it.

**4. Approach:** The approach of testing different software is different. It deals with the flow of applications for future reference. It has two aspects:

* **High-Level Scenarios:** For testing critical features high-level scenarios are written. For Example, login to a website, and book from a website.
* **The Flow Graph:**It is used when one wants to make benefits such as converging and merging easy.

**5. Assumption:**In this phase, certain assumptions will be made.

**Example:**

* The testing team will get proper support from the development team.
* The tester will get proper knowledge transfer from the development team.
* Proper resource allocation will be given by the company to the testing department.

**6. Risk:**All the risks that can happen if the assumption is broken. For Example, in the case of wrong budget estimation, the cost may overrun. Some reason that may lead to risk is:

* Test Manager has poor management skills.
* Hard to complete the project on time.
* Lack of cooperation.

**7. Mitigation Plan:**If any risk is involved then the company must have a backup plan, the purpose is to avoid errors. Some points to resolve/avoid risk:

* Test priority is to be set for each test activity.
* Managers should have leadership skills.
* Training course for the testers.

**8. Roles and Responsibilities:** All the responsibilities and role of every member of a particular testing team has to be recorded.

**Example:**

* **Test Manager:**Manages the project, takes appropriate resources, and gives project direction.
* **Tester:** Identify the testing technique, verify the test approach, and save project costs.

**9. Schedule:**Under this, it will record the start and end date of every testing-related activity. For Example, writing the test case date and ending the test case date.

**10. Defect Tracking:**It is an important process in software engineering as lots of issue arises when you develop a critical system for business. If there is any defect found while testing that defect must be given to the developer team. There are the following methods for the process of defect tracking:

* **Information Capture:**In this, we take basic information to begin the process.
* **Prioritize:**The task is prioritized based on severity and importance.
* **Communication:** Communication between the identifier of the bug and the fixer of the bug.
* **Environment:** Test the application based on hardware and software.

**Example:**The bug can be identified using bug-tracking tools such as Jira, Mantis, and Trac.

**11. Test Environments:**It is the environment that the testing team will use i.e. the list of hardware and software, while testing the application, the things that are said to be tested will be written under this section. The installation of software is also checked under this.

**Example:**

* Software configuration on different operating systems, such as Windows, Linux, Mac, etc.
* Hardware Configuration depends on RAM, ROM, etc.

**12. Entry and Exit Criteria:**The set of conditions that should be met to start any new type of testing or to end any kind of testing.

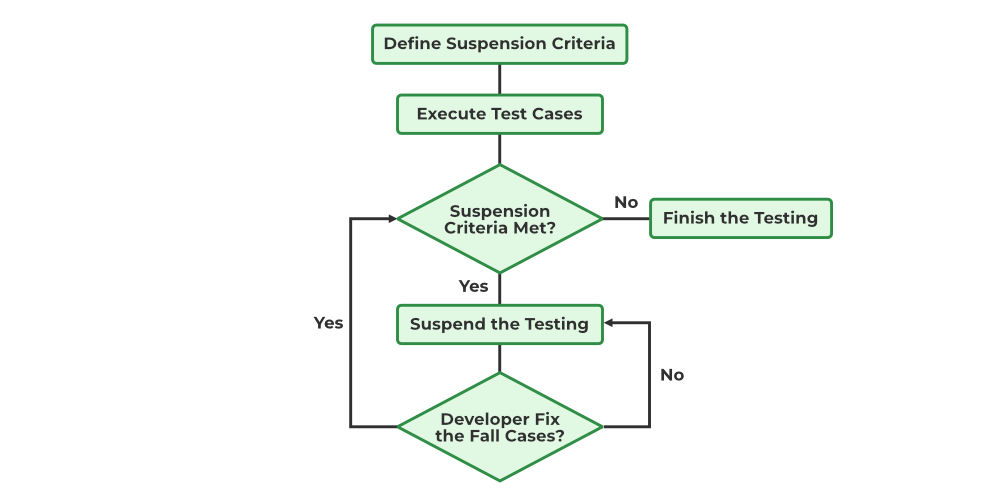
**Entry Condition:**

* Necessary resources must be ready.
* The application must be prepared.
* Test data should be ready.

**Exit Condition:**

* There should not be any major bugs.
* Most test cases should be passed.
* When all test cases are executed.

**Example:**If the team member reports that 45% of the test cases failed, then testing will be suspended until the developer team fixes all defects.



*Example of Test Plan*

**13. Test Automation:**It consists of the features that are to be automated and which features are not to be automated.

* If the feature has lots of bugs then it is categorized as Manual Testing.
* If the feature is frequently tested then it can be automated.

**14. Effort Estimation:**This involves planning the effort that needs to be applied by every team member.

**15. Test Deliverables:** It is the outcome from the testing team that is to be given to the customers at the end of the project.

Before the testing phase:

* Test plan document.
* Test case document.
* Test design specification.

During the testing phase:

* Test scripts.
* Test data.
* Error logs.

After the testing phase:

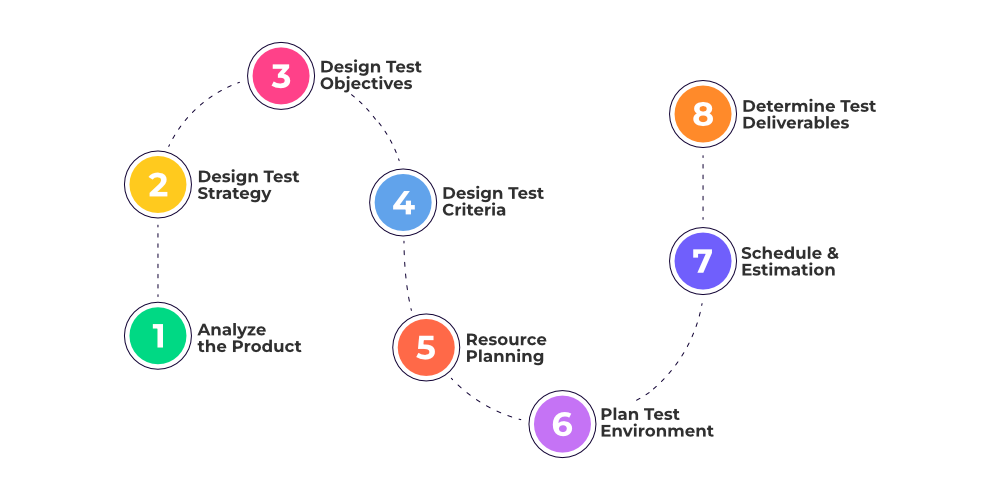
* Test Reports.
* Defect Report.
* Installation Report.

It contains a test plan, defect report, automation report, assumption report, tools, and other components that have been used for developing and maintaining the testing effort.

**16. Template:**This is followed by every kind of report that is going to be prepared by the testing team. All the test engineers will only use these templates in the project to maintain the consistency of the product.

## **How to create a Test Plan**:

Below are the eight steps that can be followed to write a test plan:



*Create Test Plan*

**1. Analyze the product:**This phase focuses on analyzing the product, Interviewing clients, designers, and developers, and performing a product walkthrough. This stage focuses on answering the following questions:

* What is the primary objective of the product?
* Who will use the product?
* What are the hardware and software specifications of the product?
* How does the product work?

**2. Design the test strategy:**The test strategy document is prepared by the manager and details the following information:

* Scope of testing which means the components that will be tested and the ones that will be skipped.
* Type of testing which means different types of tests that will be used in the project.
* Risks and issues that will list all the possible risks that may occur during testing.
* Test logistics mentions the names of the testers and the tests that will be run by them.

**3. Define test objectives:**This phase defines the objectives and expected results of the test execution. Objectives include:

* A list of software features like functionality, GUI, performance standards, etc.
* The ideal expected outcome for every aspect of the software that needs testing.

**4. Define test criteria:**Two main testing criteria determine all the activities in the testing project:

* **Suspension criteria:**Suspension criteria define the benchmarks for suspending all the tests.
* **Exit criteria:**Exit criteria define the benchmarks that signify the successful completion of the test phase or project. These are expected results and must match before moving to the next stage of development.

**5. Resource planning:**This phase aims to create a detailed list of all the resources required for project completion. For example, human effort, hardware and software requirements, all infrastructure needed, etc.

**6. Plan test environment:**This phase is very important as the test environment is where the QAs run their tests. The test environments must be real devices, installed with real browsers and operating systems so that testers can monitor software behavior in real user conditions.

**7. Schedule and Estimation:**Break down the project into smaller tasks and allocate time and effort for each task. This helps in efficient time estimation. Create a schedule to complete these tasks in the designated time with a specific amount of effort.

**8. Determine test deliverables:**Test deliverables refer to the list of documents, tools, and other equipment that must be created, provided, and maintained to support testing activities in the project.

| **Deliverables required before testing** | **Deliverables required during testing** | **Deliverables required after testing** |
| --- | --- | --- |
| Test Plan | Test Scripts | Test Results |
| Test Design | Simulators | Defect Reports |
|  | Test Data | Release Notes |
|  | Error and Execution Logs |  |

Master Software Testing and Automation in an efficient and time-bound manner by mentors with real-time industry experience. Join our [Software Automation Course](https://www.geeksforgeeks.org/courses/complete-guide-to-software-testing-automation?utm_source=geeksforgeeks&utm_medium=article_bottom_text&utm_campaign=courses) and embark on an exciting journey, mastering the skill set with ease!

# Risk Based Testing and Failure Mode and Effects Analysis

**Last Updated :**24 May, 2021

[Risk](https://www.geeksforgeeks.org/different-types-of-risks-in-software-project-development/) is the probabilityof a negative or undesirable outcome or event. Risk is any problem that might occur that would decrease customer, user, stakeholder perception of quality and/or project success.

**Types of risks:**  
There are 2 types of risks- Product Risk and Quality Risk.

1. **Product risk-**  
   When the Primary effect of a potential problem is on **product quality**, then the potential problem is called Product risk. It can also be called a quality risk. **Example-** A defect that can cause a system crash or monetary loss.
2. **Project Risk-**  
   When the Primary effect of a potential problem is onthe **overall success of the project***,* those potential problems are called Project risk. They can also be called Planning risks.  Example- Staffing issues that can delay project completion.

Not all risks are equally important. There is a number of ways we can classify the level of risk. The easiest way is to look at two factors-

1. **Likelihood**of occurrence of the problem. Likelihood arises from technical considerations.
2. **Impact**of the problem, if it occurs. Impact arises from business considerations.

**What Is Risk-Based Testing?**  
In [risk-based testing](https://www.geeksforgeeks.org/risk-management-steps-in-software-engineering/), we use the risk items identified during risk analysis, along with the level of risk associated with each risk item to guide the testing. It is a type of software testing technique that is primarily based on the probability of the risk. Risk-based testing involves the following steps:

1. Accessing the risk based on software quality.
2. Frequency of use.
3. Criticality of Business.
4. Possible areas with defects, etc.

**Characteristics Of Risk-Based Testing (RBT):**  
Below are some characteristics of Risk-based testing (RBT)-

1. RBT strategy matches the level of test effort to the level of risk. The higher the risk, the more is the test effort. This applies to test execution as well as other test activities like test designing and implementation.
2. It matches the order of testing with the level of risk. Higher risk tests tend to find more defects or are related to more important areas in the application or maybe both. So higher the risk, we plan the tests earlier in the cycle- both in design and execution. This helps in building the confidence of business stakeholders as well.
3. By effort allocation and maintaining the order of testing, the quality risk gets systematically and predictably reduced. By maintaining a traceability matrix of tests vs risks and defects identified to risks, reporting the risk as residual risks make sense. This allows the concerned stakeholders to decide when to stop testing i.e whenever the risk of continuing testing exceeds the risk of testing completion.
4. If schedule reduction requires scope reduction, it is easier to decide what needs to go out. It will always be acceptable and explainable to business stakeholders as risk levels are agreed upon.
5. To identify risks, we need to take inputs from various sources like – Requirements specifications, design specifications, existing application data, previous incident records, etc. However, if this information is not readily available, we can still use this approach by getting inputs from stakeholders for the risk identification and assessment process.

Here, Please note that the ability to sustain on little or no documentation makes this strategy of testing more robust (if not fail-proof) as dependency on upstream processes like requirement gathering is reduced to an extent.

**When To Implement Risk-Based Testing?**  
Risk-based testing approach is implemented in scenarios where-

1. There is time/resource or budget constraints. For example- A hotfix to be deployed in production.
2. When a proof of concept is being implemented.
3. When the team is new to the technology platform or to the application under test.
4. When testing in Incremental models or Iterative models.
5. Security testing is done in Cloud computing environments.

**How Risk-Based Testing Is Implemented?**  
Risk can guide testing in multiple ways but below are the major ones –

1. The effort is allocated by test managers proportionalto the risk associated with the test item.
2. Test techniques are selected in a way that matchesthe rigor and extensiveness required based on the level of risk of the test item.
3. Test activities should be carried out in reverseorder of risk i.e The Highest risk item should be tested first.
4. Prioritization and resolution of defects should be done as appropriate with the level of risk associated.
5. During test planning and control, test managers should carry out *risk control* for all significant risk items. The higher the level of risk associated, the more thoroughly it should be controlled.
6. Reporting should be done in terms of residual risks.Example- Which tests have not run yet? Which defects have not fixed yet?

It is important to note that risk management is not something that happens at the project start. It should be an **ongoing**activity throughout the project lifecycle. However, the nature of risks keeps changing depending on which test phase we are in.  
Risks should be **periodically**evaluated along with risk levels based on new developments in the project. It may result in some risks getting undervalued or even closed. Based on the outcomes, test efforts allocation and other test control activities will also change.   
Also, Effort should be to try to reduce quality risks by running the tests, finding defects, and reduce project risks by mitigating and contingency actions.

**Benefits of Risk-Based Testing (RBT):**  
By identifying and analyzing the risks related to the system, it is possible to make testing efficient and effective-

1. **Efficient-**  
   RBT is efficient because you test the most critical areas of the system early in the cycle (the earlier the defect is detected the lower is the cost of solving those issues)
2. **Effective-**  
   RBT is effective because your time is spent according to the risk rating and mitigation plans. You do not spend time on items and activities which might not be very important in the overall scheme of things.
3. **Reduced Test cases-**  
   Test case count gets reduced as test cases become more focused.
4. **Cost reduction-**  
   A reduction in cost per quality as critical issues get fixed early in the cycle and hence lowering the cost of change.
5. **Faster time to market-**  
   Faster time to market is more easily achievable with RBT approach as the most important features are available in a shippable position early in the cycle.

**Failure Mode And Effects Analysis (FMEA):**  
FMEA is a systematic technique used to identify quality risk items known as failure modes i.e.it identifies where and how the system under test might fail and then assess the relative impact of different failures.  
The steps involved in the process are:

1. **Failure modes-**  
   What could fail?
2. **Failure causes-**  
   Why would the failure happen?
3. **Failure effects-**  
   What would be the outcome of each failure?

The FMEA approach is iterative i.e re-evaluation of residual risk needs to be done repeatedly. This technique was originally designed to help prevent defects during the design and implementation phase and hence is expected to be used early in the cycle.  
There is a need to be fine-grained when it comes to failure mode analysis as effects on users, customers need to be identified as well. Since this level of depth of analysis is required, FMEA documentation can be intricate.  
It is mostly used in safety-critical, high-risk, and conservative projects. For example- Industrial control software, nuclear control software etc.   
FMEA is very useful in evaluating a new process prior to implementation and in assessing the impact of a proposed change on an existing process.

**FMEA Process:**  
This section details the FMEA process. Below are the steps-

**1. Review the Process-**Use a process flowchart to identify each process component and list each process component in the   FMEA table.  
**2. Identify potential failure modes and map their potential impact-**

1. Review existing documentation, design, and data to identify the ways each component can fail to come up with an exhaustive list.
2. Map each failure to the impact each failure has on the end product or on subsequent steps in the process.
3. Assign severity to each failure.

|  |  |
| --- | --- |
| **Severity Rank** | **Description** |
| 10 | Hazardous, without warning |
| 9 | Hazardous, with warning |
| 8 | Very High |
| 7 | High |
| 6 | Moderate |
| 5 | Low |
| 4 | Very Low |
| 3 | Minor |
| 2 | Very Minor |
| 1 | None |

**3. Once impact and severity are identified, also assign the occurrence ranking.**

1. An occurrence is a ranking number associated with the likelihood that the failure mode and its associated cause will be present in the item being analyzed. The occurrence ranking has a relative meaning rather than an absolute value and is determined without regard to the severity or likelihood of detection.
2. For System and Design FMEAs, the occurrence ranking considers the likelihood of occurrence during the design life of the product.

|  |  |
| --- | --- |
| **Occurrence Rank** | **Description** |
| 10 | >100 Per 1,000 |
| 9 | 50 Per 1,000 |
| 8 | 20 Per 1,000 |
| 7 | 10 Per 1,000 |
| 6 | 5 Per 1,000 |
| 5 | 2 Per 1,000 |
| 4 | 1 Per 1,000 |
| 3 | 0.5 Per 1,000 |
| 2 | 0.1 Per 1,000 |

1. The detection ranking considers the likelihood of detection of the failure mode/cause, according to defined criteria.
2. Detection is a relative ranking within the scope of the specific FMEA and is determined without regard to the severity or likelihood of occurrence.

**5. Calculate the Risk Priority Number.**

1. The RPN is calculated by multiplying the three scoring columns-
   1. Severity(Point2).
   2. Occurrence (Point 3).
   3. Detection (Point 4).

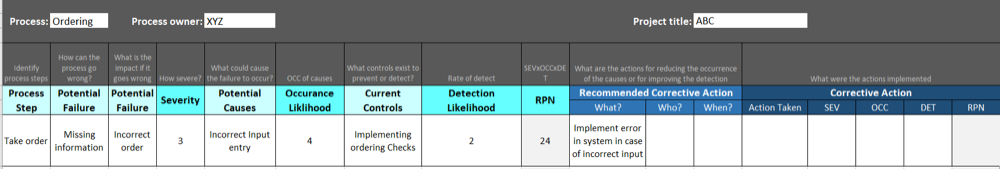
**6. Develop an action plan according to RPN.**

1. Decide which failures should be prioritized to work on based on the Risk Priority Numbers. Highest RPNs get the most priority.
2. Define action plan, Responsible person, and Expected date of completion.

**7. Implement the action plan.**

**8. Re-evaluate and repeat.**

1. Re-evaluate each of the potential failures once improvements have been made and determine the impact of the improvements.
2. Repeat the process to identify the next actions.



*FMEA Template*

**Benefits Of FMEA:**  
Below are the benefits of FMEA-

1. It is preciseand thoroughand hence less likely to omit risks.
2. It gives a holisticview of potential problems since we need to do a detailed analysis of expected system failures
3. It provides justificationfor not doing certain tests i.e where the probability of failure is least.

**Challenges Of FMEA:**  
Below are the challenges of FMEA-

1. It is documentation-heavy and hence time-consuming.
2. When trying to determine the causes of failures, it might be challenging to determine the true cause from intermediate effects.
3. Many organizations fail to recognize that the FMEA is not a static model. For successful risk management, the FMEA should be regularly updated as new potential failure modes are identified and corresponding control plans are developed.