**Gap Analysis**

A technique that businesses use to determine what steps need to be taken in order to move from its current state to its desired, future state. Also called need-gap analysis, needs analysis, and needs assessment.

Gap analysis consists of (1) listing of characteristic factors (such as attributes, competencies, performance levels) of the present situation ("what is"), (2) listing factors needed to achieve future objectives ("what should be"), and then (3) highlighting the gaps that exist and need to be filled. Gap analysis forces a company to reflect on who it is and ask who they want to be in the future.

Conducting a gap analysis can help a company re-examine its goals to determine whether it is on the right path to be able to accomplish them. A company will list the factors that define its current state, outline the factors that are required to reach the target state, and then determine how to fill the "gaps" between the two states.

**Software Testing Life Cycle**

We all must be aware of the tree’s life cycle where a small seed goes through distinct phases to gradually grow and develop as a large tree.

The similar concept of life-cycle is also followed in the software engineering field, mainly in the development life cycle and the testing life cycle where former perceives the gradual development of the business or functional requirements into a software application and the latter one visualizes the testing of the software application from a scratch to the release of the quality software application. Since, our article is not concerned to development life-cycle, we will discuss about testing life cycle only.

**What is Testing Life Cycle?**

Development life cycle is followed by the testing life cycle. A testing life cycle comprises of several phases and activities aligned in a sequential manner to initiate, execute and terminate the testing process.

A software testing process could be initiated as soon as the development process begins and may be carried out in parallel to the development activities. It can be understood through V&V development model, where a corresponding test methodology is defined for each development phase.

Now, coming back to the testing life cycle, it mainly consists of following phases in a subsequent manner.

Let’s find out what each phase consists and is responsible for.

**1. Requirement Analysis:-**

The very first phase of the software testing lifecycle involves the study and analysis of the available requirements and specifications. Both functional and non-functional requirements are being viewed and study from the testing point of view, to find out the testable requirements i.e. those requirements which may produce results on feeding with the input data.

* On the availability of requirements and specifications.
* When the application architecture is available.

**Activities**

* Brainstorming sessions for the requirement analysis and feasibility.
* Identifying and sorting out the requirement priorities.
* Creating the requirement traceability matrix (RTM).
* Identifying the suitable test environment.
* Identifying the requirements acceptable for the automated testing and the manual testing.

**Responsibility**

Requirement analysis stage visualizes the combined efforts of QA team, project manager, test manager, system architect, business analyst, client and the major stakeholders so as to have greater understanding of the requirement and subsequently the better outcomes.

**Outcomes**

* Testable Requirements.
* Requirement Traceability Matrix(RTM)
* Automation feasibility report (if applicable).

**2. Test Planning:-**

With the information gathered about the requirements in the previous phase, QA team move a step ahead in the direction of planning the testing process. Basically, a strategy or strategies is/are defined and described for the testing process/activities.

**When to go for it?**

* On the successful completion of the requirement analysis phase.
* When the testable, refined and clear requirements got defined and specified, i.e. on the availability of requirement documentation.
* Good understanding of the product domain.
* Availability of Automation feasibility report (if any).

**Activities:**

* Scope and objectives are outlined.
* Deciding the testing types to be performed along with the specific strategy for each of them.
* Roles and Responsibilities are determined and assigned.
* Identifying the resources and testing tools required for the testing.
* Estimating the time and the efforts to carry out the testing activities.
* Defining and detailing the test environment.
* Defining the time schedules.
* Entry, exit criteria along with the suspension and resumption criteria is defined.
* Planning the training activity and sessions required by the testers(if any).
* Risk analysis is being done.
* Change management process is specified and described.

**Responsibility:**

As per the requirement and the availability, QA Manager or QA lead is accountable for planning the testing process.

**Outcomes:**

* Test Plan documentation
* Time and effort estimation documentation.

**3.Test Case Design & Development:-**

The requirements has got analysed and accordingly the QA team comes out with a test plan. Now, it’s time to do some creative work and to give a shape to this test plan in the form of test cases. Based on test plan and detailed requirements, test cases are designed and developed for the purpose of verifying and validating each and every requirements specified in the documentation.

**Activities:**

* Test cases are designed, created, reviewed and approved.
* Relevant existing test cases are reviewed, updated and approved.
* Automation scripts (if any) are developed, reviewed and approved.
* Relevant test data are generated or imported from the development environment.
* Test conditions along with the input data and expect outcome for each test cases are defined and specified.

**Responsibility:**

Generally, the testers have the job of writing the test cases under the supervision of QA lead or QA manager. However, the testers may be accompanied by the developers in generating the effective automation test scripts.

**When to prepare/create test cases?**

* On the availability of software requirement specification (SRS) and business requirement specification (BRS).
* When the test plan is ready.
* Automation feasibility report(if any) is available.

**Outcomes:**

* Test cases including automation scripts.
* Test Coverage Metrics.
* Test Data

**4.Test Environment Setup:-**

The software testing process needs an appropriate platform and environment encompassing the necessary and required hardware and software, to create and replicate the favourable conditions and intended environmental factors to perform actual testing activities i.e. execution of the developed test cases on the software.

* Test data is set up.
* Test environment checklist is prepared and the required hardware and software are aggregated.
* Test server is setup and network settings are configured.
* Test Environment management and maintenance process is defined and described.
* Smoke testing of the environment to check is readiness.
* Testers are being equipped with the bug reporting tools.

**Responsibility:**

QA team under the supervision of QA manager sets up the test environment

**When to set up Test Environment?**

* When test data is ready for use.
* Test Plan documentation is available.
* Needed resources such as hardware, software, testing tools & framework, server, etc. are available.

However, the test environment set up phase may be carried out concurrently with the test case design & development stage.

**Outcomes:**

* Test Environment is set up and ready to execute tests.
* Smoke Test Results.

**5.Test Execution:-**

With the test cases, test data and the suitable test environment, QA team is now ready to try hands on some actual testing activities. The test execution phase involves the execution of the developed test cases with the help of test data in the set up test environment.

* Test Cases execution as per the test plan.
* Comparison of actual results with the expected outcomes.
* Identifying and detecting defects.
* Logging the defects and reporting the identified bugs.
* Mapping defects with the test cases and accordingly updating the requirement traceability matrix.
* Re-testing, once a defect gets fixed or removed by the development team.
* Regression testing(if required).
* Tracking a defect to its closure.

**Responsibility:**

Test Engineers are deployed to carry out the task of test case execution.

**When to go for the test execution?**

Being equipped with the test strategy, test plans, test cases, test data, properly configured and set up test environment along with some other needy resources, the QA team can kickoff the test execution process.

**Outcomes:**

* Test Status and results.
* Bug or Defect Report.
* Complete and updated Requirement Traceability Matrix (RTM).

**6.Test Closure:-**

The completion of the test execution phase and delivery of the software product marks the beginning of the test closure phase. This phase perceives the meeting and discussion amongst the QA team members with respect to test execution and its results. Apart from the test results, other testing related parameters are considered and reviewed such as quality achieved, test coverage, test metrics, project cost, adherence to deadlines, etc.

**Activities:**

* Retrospection of the whole testing process.
* Test Life Cycle exist criteria is evaluated along with some other essential aspects such as test coverage, quality achieved, fulfilment of goals and objectives, critical business goals, etc.
* Need to change the exit criteria, test strategy, test cases, etc. are discussed.
* Test Results are analysed and reviewed.
* All the test deliverables such as test plan, test strategy, test cases, etc. are collected and maintained.
* Test Closure Report and test metrics is prepared.
* Defects are arranged severity wise and priority wise.

**Responsibility:**

Generally, the QA lead or the QA Manager is responsible for preparing the test closure report.

**When to perform test closure activities?**

Generally, the test closure activity begins after the completion of test execution activities and delivery of the software product. However, it is not necessary to carry out the closure task only after the delivery of the software application. It may be performed after closure of the testing activities due to some other reasons such as achievement of targets, cancellation of the project or when the product needs update, etc.

**Outcomes:**

* Test Closure Report.
* Test Metrics.
* Learned process.

**Conclusion**

In nutshell, it may be concluded that similar to development life cycle, testing life cycle also consists of several phases and each phase counts a large number of activities to strategically and orderly carry out the testing process in an effective and efficient manner and subsequently ensuring maximum productivity and quality achievemen

**HP ALM**

HP ALM is a software that is designed to manage the various phases of Software Development Life Cycle(SDLC) right from requirements gathering to testing.

Earlier, it was known as HP Quality Center (QC). HP QC acts as a test management tool while HP ALM acts as a project management tool. HP QC is named as HP ALM from version 11.0. I am sure that this tutorial will really be a guide to those who are new to this tool.

**Advantages of using this tool**

The list given below explains the various advantages of using this tool:

* Easy to understand and easy to use.
* Provides integration with external tools such as HP UFT for automation testing and HP Load Runner for performance testing.
* Visibility of the project status to all stakeholders of the project.
* Reduces risk associated with managing several artifacts of the project at various phases.
* Reduces cost and time.
* The flexibility of usage.

**Features**

The following are the list of features provided by this tool:

* Release Management: To achieve traceability between test cases to release.
* Requirement Management: To ensure if the test cases cover all the specified requirements or not.
* Test case management: To maintain the version history of the changes done to test cases and act as a central repository for all the test cases of an application.
* Test Execution management: To track multiple instances of test case runs and to ensure the credibility of the testing effort.
* Defect Management: To ensure if the major defects uncovered are visible to all major stakeholders of the project and to make sure the defects follow a specified life cycle till closure.
* Reports Management: To ensure if reports and graphs are generated to keep a track of the project health.

**QC Versus ALM**

HP Application Lifecycle Management tool provides the core functionality of HP Quality Center along with the following features:

* Project planning and Tracking: This tool allows the users to create KPI’s (Key Performance Indicators) using ALM data and tracks them against the project milestones.
* Defect Sharing: This tool provides the ability to share defects across multiple projects.
* Project Reporting: This tool provides customized project reporting across multiple projects using pre-defined templates.
* Integration with third-party tools: This tool provided integration with third-party tools such as HP LoadRunner, HP Unified Functional Testing and REST API.

**HP ALM Version History**

HP QC was earlier known as Test Director, which was a product of Mercury Interactive. Later, Test Director was acquired by HP and the product was named as HP Quality Center.

HP Quality Center was named as HP Application Lifecycle Management from version 11.0.

**Test Case:**

A Test Case is a document that describes step by step process how to test the application. A Test Case includes Test Case ID, Steps Description, Expected Output, Actual Output, Pass/Fail, Remarks.

**1) Unit testing**: It is a test to check the code whether it is properly working or not as per the requirement.  It is done by the developers (Not testers).

2) Shakeout testing: This test is basically carried out to check the networking facility, database connectivity and the integration of modules. (It is done by the Configuration Team)

3) Smoke testing: It is an initial set of test to check whether the major functionalities are working or not and also to check the major breakdowns in the application. It is the preliminary test carried out by the SQA tester.

**4) Functional testing:**  It is a test to check whether each and every functionality of that application is working as per the requirement. It is major test where 80% of the tests are done. In this test, the Test Cases are ‘executed’.

5) Integration testing: It is a test to check whether all the modules are combined together or not and working successfully as specified in the

6) **Regression testing:** When a functionality is added to an application, we need to make sure that the newly added functionality does not break the application.  In order to make it sure, we perform a repeated testing which is called Regression Testing.  We also do regression testing after the developers fix the bugs.

7) System testing: Testing which is based on overall requirements specification and it covers all combined parts of a system. It is also a black box type of testing.

8) Load testing: It is a test to check the user’s response time of number of users using any one scenario (single business process) of the same application at the same time.

9) Stress testing: In this type of testing the application is tested against heavy load such as complex numerical values, large number of inputs, large number of queries etc. which checks for the stress/load the applications can withstand.

10) Performance testing: It is a test to check the user’s response time of number of users using multiple scenarios (multiple business process) of the same application at the same time.

**11) User acceptance testing:** In this type of testing, the software is handed over to the user in order to find out if the software meets the user expectations and works as it is expected to.

12) Black box testing: It is test where a tester performs testing without looking into the code. OR A testing method where the application under test is viewed as a black box and the internal behavior of the program is completely ignored. Testing occurs based upon the external specifications. Also known as behavioral testing, since only the external behavior of the program is evaluated and analyzed.

13) White box testing: It is a test where a tester looks into the code and performs the testing.

14) Alpha testing: In this type of testing, the users are invited at the development center where they use the application and the developers note every particular input or action carried out by the user. Any type of abnormal behavior of the system is noted and rectified by the developers.

15) Beta testing: In this type of testing, the software is distributed as a beta version to the users and users test the application at their sites. As the users explore the software, in case if any exception/defect occurs that is reported to the developers.

Reference Link:

<http://qaquestions.net/>

**Defect Management**

The Defect is an error occurred in the software any unexpected things comes with the software is called a defect. No software exists without defect or any bugs. You can’t remove the defect permanently, but you can reduce the number of defects. Defect management is a process to identify the defect of the software. The Development team needs a defect management tool so that they can find defect easily and at a very early stage of process because as soon as the defect is detected the cost of fixing it will be low, but if the defect is detected in the later stage of development process then the cost of fixing that defect will be more. Defect management will work like a backbone to developing a team in finding the defect in the early stage in a very easy way. The small investment in defect management tool will give you the greatest return. Defect management works in the parallel way of software development process. Software testing team finds all the bugs and defect and report to developing a team to fix it. Defect management, records all the defect of the software these records can be seen later as well if you want to review or want to check that how you have fixed that. These records can be used the knowledge-base for the future reference. Example like XYZ company is making software, but they are not using a issue tracking system, then finding the defect in the software will be very difficult and complex task and if they are not able to find at an early stage, then the cost fixing it at a later stage will cost more, but if they invest in the defect management tool than these defects can be found easily at very early stage and all the defect are recorded in the tool and can be used for future reference. Small invest can give more benefits.

**Advantages of Defect Management**

1. There is no shortage of automation tools when it comes to defect tracking. You can find the software tools to track the non-technical problems, user-facing tools for production-related defects, & internal automated tools the development team can use to track the defects. In fact, even if you are just using the email, spreadsheets, sticky notes, and a log on a wiki to track the customer’s issues, you will need the defect tracking of few slots. It is just a matter of figuring out the correct tools & activities for the team.

2. The defect tracking assists you to ensure that the bugs found in the system really get fixed. Sure, it is great for developers and testers have a conversation & to recreate the issue together.

3. In defect management, the defect tracking, automated tools not only provide a channel to ensure follow through but also provide the valuable-metrics. Depending on application tools being used, the teams can tie the defects for changed the code, tests, and the other data that’ll allow for traceability and analysis on the defect trends. If the certain module strains with the defects, it might be time for review & rewrite that module.

4. Having the tool in spot also sends notifications for the right individual when the bug needs to be tested, fixed, and marked as resolved. In defect management tools , the defect tracking app tools, it allows for a repository of the documentation that’ll provide the value for troubleshooters and for support the workers later on if there is a workaround to a problem.

**Defect management process**

**Identification:** This will be the first step to identify the defect in the process this can be done by the testing team and sometimes the customer can also tell you about the defect.

**Categorization:** After the defect is registered, then it passes to the desired person then checks it and categorize it that what type of defect it is and then move to the next step.

**Prioritization:** According to the type of defect or severity of the defect. These defects are set of priorities for fixing it accordingly. It can handle through a formal channel or an individual team member who is working on defect can set its own priorities for fixing defects.

**Assignment:** As soon as the priorities are decided by formal channel defect is assigned to a developer to fix it.

**Resolution:** Developer fixes the defect and places it in the same place from where it was identified.

**Verification:** As soon as the defect is fixed, the developing team verifies that defect is actually fixing and working in the smoothest way or not.

**Closure:** once it is fixed and resolved it is marked as closed case.

**Management Reporting:** This provided at regular intervals which helps management to know which work is done and which work is going on and when it will be done.

**Manual VS Automation**

Software testing is a huge domain, but it can be broadly categorized into two areas: manual testing and automated testing.

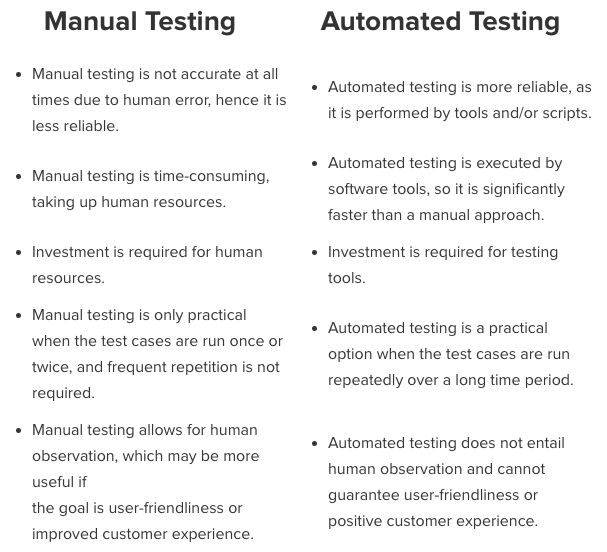
Both manual and automated testing offer benefits and disadvantages. It’s worth knowing the difference, and when to use one or the other for best results.

In manual testing (as the name suggests), test cases are executed manually (by a human, that is) without any support from tools or scripts. But with automated testing, test cases are executed with the assistance of tools, scripts, and software.

Testing is an integral part of any successful software project. The type of testing (manual or automated) depends on various factors, including project requirements, budget, timeline, expertise, and suitability. Three vital factors of any project are of course time, cost, and quality - the goal of any successful project is to reduce the cost and time required to complete it successfully while maintaining quality output. When it comes to testing, one type may accomplish this goal better than the other.

**Manual vs. Automated Testing: the Pros and Cons**

Manual testing and automated testing cover two vast areas. Within each category, specific testing methods are available, such as black box testing, white box testing, integration testing, system testing, performance testing, and [load testing](https://www.apicasystem.com/load-testing/). Some of these methods are better suited to manual testing, and some are best performed through automation. Here’s a brief comparison of each type, along with some pros and cons:

****

**When Should I Use Manual vs. Automated Testing?**

In short, manual testing is best suited to the following areas/scenarios:

* Exploratory Testing: This type of testing requires the tester’s knowledge, experience, analytical/logical skills, creativity, and intuition. The test is characterized here by poorly written specification documentation, and/or a short time for execution. We need the human skills to execute the testing process in this scenario.
* Usability Testing: This is an area in which you need to measure how user-friendly, efficient, or convenient the software or product is for the end users. Here, human observation is the most important factor, so a manual approach is preferable.
* Ad-hoc Testing: In this scenario, there is no specific approach. It is a totally unplanned method of testing where the understanding and insight of the tester is the only important factor.

Automated testing is the preferred option in the following areas/scenarios:

* Regression Testing: Here, automated testing is suitable because of frequent code changes and the ability to run the regressions in a timely manner.
* Load Testing: Automated testing is also the best way to complete the testing efficiently when it comes to load testing. Learn more about load testing with our best practices guide.
* Repeated Execution: Testing which requires the repeated execution of a task is best automated.
* Performance Testing: Similarly, testing which requires the simulation of thousands of concurrent users requires automation.

Keeping these factors in mind, you can find the best approach in any given testing situation and achieve quality output well within your budget and timeline.