Test Lead Basics Manual Testing Sample Interview Question and Answers prepared by Haradhan Pal

YouTube Channel Link: https://www.youtube.com/c/HaradhanAutomationLibrary?sub_confirmation=1

Project status reporting on a daily basis to onsite counterparts and client.

Daily Status Report

- Progress Highlight
- Blocker Status(Defect, Test Data, Clarification, Requirement update)
- Summary Test Execution(Failed, Passed, Blocked, NC %
- Test Execution GOAL(Effort Justification)
- Outage Details(Resource #/ Effort Loss/Person, Planned productive Hours/Actual productive hours/Effort Utilized in other Area)
- Test Case and defect dump.
- Resourcing

Difference between Test Strategy and Test Plan

Test Strategy

A Test Strategy document is a high level document and normally developed by project manager. This document defines "Software Testing Approach" to achieve testing objectives. The Test Strategy is normally derived from the Business Requirement Specification document.
The Test Strategy document is a static document meaning that it is not updated too often. It sets the standards for testing processes and activities and other documents such as the Test Plan draws its contents from those standards set in the Test Strategy Document.
Some companies include the "Test Approach" or "Strategy" inside the Test Plan, which is fine and it is usually the case for small projects. However, for larger projects, there is one Test Strategy document and different number of Test Plans for each phase or level of testing.
Components of the Test Strategy document

- Scope and Objectives
- Business issues
- Roles and responsibilities
- Communication and status reporting
- Test deliverables
- Industry standards to follow
- Test automation and tools
- Testing measurements and metrics
- Risks and mitigation
- Defect reporting and tracking
- Change and configuration management
- Training plan

Here is an example of an **Agile Test Strategy Document Template**

Test Plan

The Test Plan document on the other hand, is derived from the Product Description, Software Requirement Specification SRS, or Use Case Documents.
The Test Plan document is usually prepared by the Test Lead or Test Manager and the focus of the document is to describe what to test, how to test, when to test and who will do what test.
It is not uncommon to have one Master Test Plan which is a common document for the test phases and each test phase have their own Test Plan documents.
There is much debate, as to whether the Test Plan document should also be a static document like the Test Strategy document mentioned above or should it be updated every often to reflect changes according to the direction of the project and activities.

My own personal view is that when a testing phase starts and the Test Manager is “controlling” the activities, the test plan should be updated to reflect any deviation from the original plan. After all, Planning and Control are continuous activities in the formal test process.

Components of the Test Plan document

- Test Plan id
- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Test techniques
- Testing tasks
- Suspension criteria
- Features pass or fail criteria
- Test environment (Entry criteria, Exit criteria)
- Test deliverables
- Staff and training needs
- Responsibilities
- Schedule

Test matrices formula

Effort Variance %	Effort Variance = (Actual Effort (Hrs.) - Estimated Effort (Hrs.)) / (Estimated Effort (Hrs.)) * 100
Onsite-Offshore Ratio	Onsite Ratio = (Total Onshore Associates / Total resources) * 100 Offshore Ratio = (Total Offshore Associates / Total resources) * 100
QA Cost as % of Total IT Cost	QA Cost as a % of Total IT Cost = Total QA Cost / Total IT Cost * 100
QA Effort as % of Total IT Effort	QA Effort as a % of Total IT Effort = Total QA Effort / Total IT Effort * 100
Automation Effort Saving %	Regression Automation Effort Savings % = (Actual Effort for Manual test execution - Actual Effort for Automated test execution) / Actual Effort for Manual execution * 100%
Defect Leakage	Defect Leakage = ((Number of defects unidentified during testing phase but found in production / (Number of defects unidentified during testing phase but found in production + Total number of valid defects identified during testing phase)) * 100
Test Effectiveness	Test Effectiveness = ((Number of defects identified during QA testing phase - Number of invalid defects) / (Number of defects identified during testing phase + Total number of valid defects in UAT phase)) * 100
Test Case Pass Rate %	Cumulative Test Case Pass Rate % = (Total Number of Test Cases Passed / Total Number of instances executed [First Run + Repeat]) * 100
Requirement Stability Index	Requirement Stability Index = (Number of Original Baseline Test conditions + Number of Baseline Test conditions changed + Number of New Test conditions added to Baseline Test conditions + number of baseline Test conditions deleted) / (Number of Original Baseline Test Conditions)
Defect Reopen Rate %	Defect Re-Open Rate = (Total Number of times defects re-opened / Total Number of times Defects Fixed) * 100
Defect Discovery Rate %	Defect Discovery Rate = (Total number of defects found / Number of Test Cases executed) * 100
Test Design Coverage %	Test Design Coverage = (Number of requirements covered in test design by QA /Scope of the Requirements to be covered as mentioned in Test plan (after finalizing any Risk Based approach) * 100
Test Execution Coverage %	Test Execution Coverage = (Total number of Planned Test Case Executed (Pass+ Fail+ Blocked) / Total Number of Planned Test Cases) * 100
Schedule Compliance %(Applicable for Bond)	Schedule Compliance = (Total number of requests not available for UAT / Total number of requests for the release) * 100
Schedule Variation %(All other LOB's)	Schedule Variation = Network Days of ((Actual End Date - Planned End Date) / (Planned End Date-Planned Start Date)) till the reporting period *100
Test Design Productivity- TCP	Test Design Productivity (TCP) = ((Number of TCP Created) / (Total Effort spent for Test design activity))

Test Execution Productivity- TCP	Test Execution Productivity (TCP) = ((Number of TCP Executed) / (Effort spent for Test Case Execution))
Regression Automation %	Regression Automation % = (Total Number of Regression Test Cases Automated / Total Number of Regression Test Cases) * 100

Test Reports

1. Daily Status Report
2. Predictive Dashboard Report
3. Test Closure Report
   (Purpose/Approach/TM Tool/Inscope-outscope/Requirement Traceability/Test Schedule/Test Summary Highlights/Test Result Details/Lesson Learnt/Recommendations/Test Metrics /Risk/Test Deliverables)

Defect Severity & Priority

1) Severity:
It is the extent to which the defect can affect the software. In other words it defines the impact that a given defect has on the system. **For example:** If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.
Severity can be of following types:

- **Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
- **Major:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
- **Moderate:** The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.
- **Minor:** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
- **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

2) Priority:
Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait? This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements. **For example:** If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.
Priority can be of following types:

- **Low:** The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect have been fixed.
- **Medium:** The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
- **High:** The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.

Few very important scenarios related to the severity and priority which are asked during the interview:
High Priority & High Severity: An error which occurs on the basic functionality of the application and will not allow the user to use the system. (E.g. A site maintaining the student details, on saving record if it, doesn't allow to save the record then this is high priority and high severity bug.)
High Priority & Low Severity: The spelling mistakes that happens on the cover page or heading or title of an application.
High Severity & Low Priority: An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.

Low Priority and Low Severity: Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

Effort Estimations

Testing scope || Availability of Resources || Type of Testing || No of Test Cycle || Failure Rate (10% to 20%) || Productivity || Contingency Factor || Non Testing Activities || Buffer time for Resource onboarding

Risk management

The process of identifying the risk (Product risk or Project risk), analyzing the risk and then mitigating the risk or controlling the risk is known as Risk Management.

The risk management process occurs twice, during:

1. Test planning
2. Test case design(end) or sometimes in the test execution phase

   It is mandatory in case 1 but with case 2 it is more of a 'need-basis' situation

Risk Management Process

The generic process for Risk management involves 3 important stages:

1. Risk identification
2. Risk impact analysis
3. Risk mitigation

Risk identification

As it is said, the first step to solving a problem is identifying it. This stage involves making a list of everything that might potentially come up and disrupt the normal flow of events.

The main outcome of this step is: a list of risks.

This risk based testing step is commonly led by the QA lead/Manager/representative. However, the lead alone will not be able to come up the entire list- the entire QA team's input makes a huge impact. We can say this is a collective activity led by the QA lead.

Also, the risks that are identified during the Test planning phase are more 'managerial' in orientation- meaning, we are going to look at anything that might impact the QA project's schedule, effort, budget, infrastructure changes, etc. The focus here is not the AUT, but the way the QA phase will go on.

Risks during Test Planning – Risk Based Testing Examples

The following is a sample list of risks that might be listed during test planning phase. Please note, that the AUT and its functionality is not the focus here.

#1. Testing schedule is tight. If the start of the testing is delayed due to design tasks, the test cannot be extended beyond the UAT scheduled start date.

#2. Not enough resources, resources on boarding too late (process takes around 15 days.)

#3. Defects are found at a late stage of the cycle or at a late cycle; defects discovered late are most likely be due to unclear specifications and are time consuming to resolve.

#4. Scope not defined completely defined

#5. Natural disasters

#6. Non-availability of Independent Test environment and accessibility

#7. Delayed Testing Due To new Issues

At this point, you can choose to be as thorough as you would like, depending on the amount of time available.

Once, all the risks are listed, we progress to Risk assessment/Risk impact analysis.

Risk assessment/Risk impact analysis

Risk Analysis in software testing: All the risks are quantified and prioritized in this step. Every risk's probability (the chance of occurrence) and impact (amount of loss that it would cause when this risk materializes) are determined systematically.

High – medium – low, values are assigned to both the probability and impact for each risk. The risks with "high" probability and "High" impact are taken care of first and then the order follows.

Risk impact analysis table:

Following these steps, the risk impact analysis table for the above risks listed would look something like this (all values are hypothetical and only for understanding purposes):

Risk	Probability	Impact
1. Testing schedule is tight. If the start of the testing is delayed due to design tasks, the test cannot be extended beyond the UAT scheduled start date.	High	High
2. Not enough resources, resources on boarding too late (process takes around 15 days.)	Medium	High
3. Defects are found at a late stage of the cycle or at a late cycle; defects discovered late are most likely be due to unclear specifications and are time consuming to resolve.	Medium	High
4. Scope not defined completely defined	Medium	Medium

Risk	Probability	Impact
5. Natural disasters	Low	Medium
6. Non-availability of Independent Test environment and accessibility	Medium	High
7. Delayed Testing Due To new Issues	Medium	High

Risk Mitigation

The final step in this Risk Based Testing (RBT) process is to find solutions to plan how to handle each one of these situations. These plans can differ from company to company, project to project and even person to person.

Risk Mitigation Techniques:

Here is an example of what the Risk's table transforms into when this phase is complete:

Risk	Prob.	Impact	Mitigation Plan
SCHEDULE Testing schedule is tight. If the start of the testing is delayed due to design tasks, the test cannot be extended beyond the UAT scheduled start date.	High	High	• The testing team can control the preparation tasks (in advance) and the early communication with involved parties. • Some buffer has been added to the schedule for contingencies, although not as much as best practices advise.
RESOURCES Not enough resources, resources on boarding too late (process takes around 15 days.	Medium	High	Holidays and vacation have been estimated and built into the schedule; deviations from the estimation could derive in delays in the testing.
DEFECTS Defects are found at a late stage of the cycle or at a late cycle; defects discovered late are most likely be due to unclear specifications and are time consuming to resolve.	Medium	High	Defect management plan is in place to ensure prompt communication and fixing of issues.
SCOPE Scope completely defined	Medium	Medium	Scope is well defined but the changes are in the functionality are not yet finalized or keep on changing.
Natural disasters	Low	Medium	Teams and responsibilities have been spread to two different geographic areas. In a catastrophic event in one of the areas, there will resources in the other areas needed to continue (although at a slower pace) the testing activities.
Non-availability of Independent Test environment and accessibility	Medium	High	Due to non-availability of the environment, the schedule gets impacted and will lead to delayed start of Test execution.
Delayed Testing Due To new Issues	Medium	High	During testing, there is a good chance that some "new" defects may be identified and may become an issue that will take time to resolve. There are defects that can be raised during testing because of unclear document specification. These defects can yield to an issue that will need time to be resolved. If these issues become showstoppers, it will greatly impact on the overall project schedule. If new defects are discovered, the defect management and issue management procedures are in place to immediately provide a resolution.

A few points to note:

1. The sooner risk management starts in a QA project planning phase, the better.
2. Of all 3 steps, **risk identification is the most important**. If anything is not listed and considered for further steps, the risk goes unhandled.
3. Try to find an ideal time frame for this activity. Remember, too much planning leaves too little time for doing.
4. Also, after the risk management process, if a new situation comes up, the risk management plan can be altered or updated to reflect the most current condition.
5. **Historical data** can be very useful for the success of this process.

Risk Based Testing

Risk Based Testing (RBT) is a testing process with unique features. It is basically for those project and application that is based on risk. Using risk, Risk based testing prioritize and emphasize the suitable tests at the time of test execution. In other word, Risk is the chance of event of an unwanted outcome. This unwanted outcome is also related with an impact. Some time it is difficult to test all functionality of the application or it is might not possible. Use Risk based testing in that case; it tests the functionality which has the highest impact and probability of failure.

It's better to start risk based testing with product risk analysis. There are numerous methods used for this are,

- Clear understanding of software requirements specification, design documents and other documents.
- Brainstorming with the project stakeholders.

Risk-based testing is the process to understand testing efforts in a way that reduces the remaining level of product risk when the system is developed,

- Risk-based testing applied to the project at very initial level, identifies risks of the project that expose the quality of the project, this knowledge guides to testing planning, specification, preparation and execution.
- Risk-based testing includes both mitigation (testing to give chances to decrease the likelihood of faults, specially high-impact faults) and contingency (testing to know work-around to create the defects that do get past us less painful).
- Risk-based testing also includes measurement process that recognizes how well we are working at finding and removing faults in key areas.
- Risk-based testing also uses risk analysis to recognize proactive chances to take out or avoid defects through non-testing activities and to help us select which test activities to perform.

Major processes to execute the Risk-based testing are described below:

- Process 1 – Describe all requirements in terms of Risk involved in the project
- Process 2 – In terms of risk assessment, prioritize the requirements
- Process 3 – Plan and define tests according to requirement prioritization
- Process 4 – Execute test according to prioritization and acceptance criteria.

HPQC Reports

Report Category	Reports Included in the HP Quality Center Report Packs		
	Free	**Standard**	**Professional**
Test Management	Test Plan	Test Plan	Test Plan, Test Steps, Test Coverage Gap
Test Execution Management	Test Lab, Test Run Progress	Test Lab, Cycle Status, Failed Test, Project Cycles, Test Run Progress	Test Lab, Test Result History, Cycle Status, Failed Test, Project Cycles, Test Run Progress
Requirement Management	Requirement Status	Requirement Status	Requirement Priority, Traceability Matrix, Requirement Status
Defect Management	Defect Assignee	Defect Triage, Not Verified Defects, Defect Assignee	Defect Triage, Not Verified Defects, Defect Assignee
Team Productivity	Team Progress	Team Progress	Team Progress

HPQC Tabs

Dashboard (Analysis & Dashboard) // Management (Release) // Requirements //Testing (Test Plan & Test Lab) // Defects

Regression testing

Regression testing is a type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system after changes such as enhancements, patches or configuration changes, have been made to them.

Regression indicator (New/Mod/Beginning) /Regression Scope (Mandatory/Optional)/ Regression priority (High/Medium/Low)

1. End To End Regression
2. Application Specific Regression

End To End testing

End-to-end testing is a methodology used to test whether the flow of an application is performing as designed from start to finish. The purpose of carrying out end-to-end tests is to identify system dependencies and to ensure that the right information is passed between various system components and systems.