| [Print](javascript:window.print()) | [Close](javascript:parent.window.close()) |

**Course Transcript**

Software Test Management

**Test Organization and Independence**

| [1. Dependent and Independent Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#t2) |

| [2. Tasks of the Test Leader and Tester in Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#t5) |

| [3. Organizing a Software Test Team](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#t8) |

**Test Planning and Estimation**

| [1. Software Test Planning](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#t12) |

| [2. Test Estimation, Test Strategies, Entry and Exit Criteria](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#t15) |

| [3. Software Test Progress Monitoring and Control](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#t18) |

| [4. Test Reporting and Summarization](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#t21) |

Dependent and Independent Software Testing

Learning Objective

*After completing this topic, you should be able to*

* *recognize a test team's level of independence in an organization*

**1. Levels of independence**

The software testing activity may take on one of many forms within an organization from being fully integrated with the development life cycle to the complete outsourcing of responsibilities to a third-party. An organization uses these styles of testing based on its testing requirements and approach towards testing.  
  
The two testing styles that an organization can use are the dependent and independent styles.   
  
Dependent testing involves developers testing the software they create while independent testing sees someone other than the developers testing the software.

Independent testing can be more beneficial than dependent testing, because it adds a level of objectivity to the testing process. For example, developers who create the application may not be able to identify defects because they are familiar with the software application. A tester that is independent from the application is more likely to be critical of the application and therefore find more errors and report them with supporting evidence.

Organizations may also employ different levels of independence within the independent testing process. These levels of independence are inversely proportional to the level of association a tester has with the application.

Graphic

*Description of independence level line graph:  
  
The chart shows how the Level of independence increases with the decrease in the level of association of a tester with an application.  
  
Description ends.*

**Developers**

The lowest level of independence is employed when a software developer within the development team tests the software.  
  
Suppose Sandra has developed a word processor application. Brian, a fellow developer, performs a peer review on her application and detects minor errors. This testing is not sufficient to uncover all defects in the application because the creator of the application and its tester are from the same functional unit. Because of this, the level of independence employed is low.

**Testers within a development team**

When integrated testers who work alongside developers review the software product, the level of independence employed is slightly higher.  
  
In this situation, when you develop an application and want to test specific modules, you hand over the modules to the integrated test team. These testers, who report to the development manager, check the modules. The level of independence in such a situation is low because the testers work in close collaboration with the developers, which reduces the objectivity in testing.

**Testers within an organization**

You implement a slightly higher level of testing independence by using the test team within your organization to review your software products.  
  
Using an internal QA team is common in most development environments. However, this independence level is only marginally higher than using a developer or an integrated tester. The testers and the developers in this case are within the same organization, and are not completely independent of one another.

**Testers provided by operational business units**

When a tester from an operational business unit is employed to test your software application, the level of independence is increased further. In this case, the separation of responsibilities between the testers and the developers is more evident.  
  
For example, independent testing units within the same organization can test both client and in-house software.

**Test specialists**

To implement a highly independent test practice, you employ specialist testers, such as usability, security, and certification testers.  
  
Suppose your team has developed a web service that is built on a recently updated development environment. To test the service, you can hire a certified tester who is competent in the new environment. This would ensure a highly independent and effective test approach for your application.

**Outsourced testers**

The highest level of testing independence is implemented when a third-party testing team tests your application. Although this approach is expensive, it ensures the highest degree of independence.  
  
For example, after you develop an application, you can outsource it to third-party testers. They would perform a complete review of the application and provide objective feedback.

When you apply the independent testing approach in your organization, benefits include

**difference in perspective**

Independent testing provides a different set of testing perspectives. Independent testers view the product from their perspective instead of the developers'. Because the testers are unbiased, they can identify varied defects that developers can miss. The perspective of a developer reflects that of a creator, who is more likely to be possessive and biased regarding the application. Due to this, developers might overlook critical defects in functionality.

**verification of assumptions**

Independent testers verify the assumptions made during the specification and implementation phases of the test process. This prevents them from logging invalid defects against the application and missing critical errors. Suppose you've made an assumption that your airline ticket software application doesn't allow you to book domestic air tickets. If independent testers are aware of this assumption, they would not log defects that are not part of the testing domain.

**different sets of assumptions**

Independent testers make assumptions that are different from those of developers; this helps them to detect hidden problems in the application. For example, developers may understand the working model of the application they develop and make a number of assumptions regarding its performance such as, only 75% of the application's functionality is measurable. But the independent tester being objective to the product makes a completely different set of assumptions and might expect to measure 100% of the application's functionality for performance.

Although independent testing is beneficial for an organization, it does have its drawbacks. These may include

**isolation**

Independent testing is performed in isolation from the development team. Because of this, testers might have to rely on test cases that are rarely updated to communicate the scope of testing. For example, testers and developers of the same project working from different geographical locations might face difficulties in communicating the details of testing.

**lack of communication**

With independent testing there may exist a lack of communication between developers and testers. Suppose a testing team requires clarification about an application's functionality. They could either read the test cases, or ask the developer to clarify the doubts. But if the communication process between the testers and developers is not strong, the parties might not be able to arrive at a solution promptly. Unfortunately, this may introduce unexpected delays in the development of the product.

**probable delay in delivery**

Independent testing is time-consuming and may delay project delivery. This testing is usually done in the final stage of the project, so it may become a bottleneck if there were delays at earlier stages. For example, developers might refuse to fix a number of defects logged by independent testers because of difference in perspectives. This results in repeated rounds of testing and fixing, which delays delivery.

**less responsibility**

Independent testing reduces the sense of responsibility in the development team. For example, developers might submit software with known errors to the test team, knowing that existing errors take time to fix. In this case, the developers don't aim for zero-defect products, and they wait for the test team to report the errors.

In addition to the dependent and independent approaches, there is one more approach to software testing – the agile approach. In this approach, quality assurance is more integrated with development than in more "traditional" development models, where everybody performs multiple roles. Because of this, the level of independence is comparatively lower than other testing approaches. Peer reviews are an example of this approach; developers working in the same team test the applications developed by their peers.  
  
The agile testing approach ensures minimal overhead on communication and coordination, and developers are aware of their responsibility for quality. Additionally, the technology and application expertise required for quality assurance tasks are similar to the expertise required for development tasks. So a developer can perform testing, even if he or she has no specific expertise in certain skills, such as test design.

Question

As a software developer, the applications that you develop are tested by a team of testers that works with you but reports to the development manager. Identify the level of independence of the testing team.

**Options:**

1. No independence
2. Slightly higher level of independence
3. Highest level of independence

Answer

***Option 1:****Incorrect. There is no independence when the developer who has created the application tests it.*

***Option 2:****Correct. The independence level is slightly higher when testers within the development team reporting to the development manager, or outside the team reporting to the project manager, perform testing.*

***Option 3:****Incorrect. There is complete independence when testing is performed by an outsourced test team.*

**Correct answer(s):**

2. Slightly higher level of independence

**2. Summary**

There are two types of testing – dependent and independent. Dependent testing involves developers testing their own applications while independent testing involves someone other than the developers testing the applications.  
  
There are also different levels of independence within the independent testing process. These levels of independence are inversely proportional to the level of association a tester has with the application.  
  
Independent testing is beneficial to an organization because those who perform the testing bring a different set of testing perspectives and assumptions to the process. Because of this independence, they also verify the assumptions made by developers. However, there are also disadvantages of independent testing. Independent testing is often done in isolation from the development process and may introduce challenges that could end up delaying the testing and hence, the development process.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#top)

Tasks of the Test Leader and Tester in Software Testing

Learning Objectives

*After completing this topic, you should be able to*

* *recognize the tasks of a test leader*
* *recognize the tasks that are carried out by a tester*

**1. Tasks of a test leader and tester**

A test team consists of a number of resources who perform a wide variety of roles. Some of these roles may not necessarily be utilized in many organizations. However, the roles that are common across test teams – regardless of the organization – are those of a test leader and a tester. A test leader is a skilled professional who manages the testing activities and its resources. The person who performs testing – such as system or component testing – is known as a tester.

These testing roles can be performed by anyone who has the skills and aptitude for software testing. It is not mandatory that testers be certified professionals. For example, a team leader who understands the test process can perform both roles, based on the project and organizational requirements. Factors such as resource requirements and project-delivery life cycle affect the distribution of roles in testing.

While analyzing test roles and activities, you should note that a test role is not the same as a test job. A role can be defined as one or many responsibilities to which an individual is assigned. An example of this is the role of a test leader. On the other hand, a test job is what an individual is employed to do. This job may consist of one or more roles. For example, a tester can also be a test leader.

Test leaders manage the test team. They plan, monitor, and implement the testing activities and tasks. They can also be test managers, test coordinators, test team leaders, and test program managers.

The tasks of a test leader are similar to those of a project manager. Because of this, a project manager, a development manager, or a quality assurance manager may perform the tasks of a test leader, depending on project requirements. However, in these cases, the level of independence is low. Also, non-testers who lead the test team may not have the knowledge and the skill set required to manage testing activities. Test leaders must have the appropriate mindset and training to plan, monitor, and control testing effectively.

Tasks that a test leader performs are

**creating test policies and strategies**

Test policy contains the rules of testing. For example, your software product is completely driven by its design statements. In this case, as a test leader, you define a test policy that states the testers must always review the design documents before testing the product.  
  
Test strategy provides a high-level overview of the methods used to test the software. These test strategies are based on the approach you'd adopt for testing your product. For example, if an independent test team decides to perform system testing for a software product, then you coordinate with the project managers and other stakeholders and create test strategies that match the system-testing approach.

**selecting the testing tools and setting up the environment**

The test leader determines the tools required for testing. These include spreadsheets, word processors, project planning tools, and non-IT tools and equipment. If required, the test leader should make arrangements for training the test team members for using these tools. For example, if your organization uses a defect-logging tool for the test team, as a test leader, you should ensure that the testers in your team are trained to use this tool.  
  
Additionally, the test leader decides the structure and implementation of the test environment. The test environment consists of the supporting hardware and software required for running the tests.

**planning and managing the tests**

The test leader selects appropriate test approaches; estimates the time, effort, and cost of testing; acquires resources; defines test levels and cycles; and plans incident management. The test leader also manages the specification, preparation, implementation, and execution of tests. To analyze test specifications, the test leader analyzes, reviews, and assesses user requirements, design specifications, and test models.  
  
For example, the test leader should determine the number of testers and the hours of effort required to test a specific application.

**ensuring configuration management**

Configuration management means identifying the items that make up software or comprise a system and ensuring that these items are managed carefully throughout the project and product life cycle. The test leader ensures appropriate configuration management of the testware created and traceability of the tests.  
  
For example, to test a web service, you would need to analyze the elements that constitute this application, such as the framework on which the application is built and its development environment.

Additional tasks of a test leader include

**planning and implementing automation**

Test leaders decide the activities that need to be automated, and the degree and method used for automation. They also ensure that automation is implemented according to the plan.  
  
Suppose you want to implement partial automation for testing a software product. To do this, you identify the specific modules of the software that need to undergo automated testing and assign appropriate strategies.

**preparing test schedules**

The test leader prepares a schedule for all testing-related activities. This helps track the activities and ensure completion of testing on time. The schedule includes the details of the test analysis and design tasks, test implementation, execution, and evaluation.  
  
For example, the test leader would draw up a test schedule with planned start and end dates for each of the test activities to be performed.

**assisting other project activities**

The test leader sometimes provides testing perspective to other project activities, such as integration, planning, and scheduling.  
  
Suppose unforeseen delays have occurred during the development phase of the project and you suspect that this would affect the testing schedule. As a test leader, you would then need to discuss the development schedule with the project manager and ensure that the testing activities are executed seamlessly.

**measuring test progress**

The test leader compares actual test progress with planned progress. If there are any changes or deviations, the test leader takes appropriate action to meet the objectives of the project.  
  
For example, after one module of testing is complete, the test leader can estimate the overall progress of the test process for all modules. If the overall progress of testing activities is not at the planned level, the test leader could then employ additional testers to complete the required testing activity on time.

**preparing test summary reports**

During the closing stages of the project, the test leader prepares the test summary reports based on the information gathered during testing. This helps in making informed decisions about the software. The outcome of the test should be known not only to the testers but also to the stakeholders.  
  
The test summary report could include the number of testers deployed to complete testing, the total number of hours spent, and the number of defects detected in the application.

In addition to the test leader role, the other common test role is that of a tester. Some of the basic qualifications that a tester must have include the ability to communicate effectively and prepare and present reports efficiently.

In addition to the basic qualifications for testing, testers should also acquire skills in other areas.

**Application**

Testers should be aware of the specifications, benefits, and limitations of the application being tested, as well as the technology on which the application is built. This helps in identifying the functions and features that are prone to defects.  
  
Suppose you are testing a SharePoint services application. Before executing the test suites, you should thoroughly analyze the user requirements and the drawbacks of using a SharePoint site, so that you can identify valid defects.

**Technology**

A tester must be able to determine the important functions and features of the system that is tested, its intended behavior, the issues that it resolves, and the process that it automates.  
  
To illustrate this concept, if you were testing a web application, you would analyze the framework and the integrated development environment with which the application is designed and developed, for example, Visual Studio.

**Testing**

Being accountable for the test execution, testers should be experts at all the test processes and techniques used to evaluate software.  
  
Suppose a new test technique has been identified to evaluate the .NET-based web services. Testers who test these type of applications should be aware of and be able to use any new techniques assigned for this test.

The type and the level of skill required in each area depends on the project, organization, application, and risks involved. Because a variety of testing tasks exist, the knowledge and skills required for performing these tasks are varied. You can often find people performing specialized tasks. For example, database experts perform database testing, while security experts test security-related applications.

The tasks of a tester include

**creating test plans and test designs**

Developing and reviewing test plans are the prime tasks of a tester. The test plan provides an outline of the testing process. Additionally, the tester identifies test conditions and creates test designs, test cases, test procedure specifications, and test data.  
  
For example, in an auction web page that you want to test, you want users to be able to view automobiles and place bids on the web site. You can set these requirements as test conditions, to check if the web page functions as expected to display the selected automobile.

**analyzing test specifications**

The tester needs to analyze, review, and assess the specifications of the applications, such as user requirements, design specifications, and test models.  
  
For example, to test a messenger service application, you would need to analyze the user profile and the specifications of the application, such as its look and feel, updated functionality, and compliance with other applications.

**setting up the test environment**

Setting up the test environment or coordinating with the system administration and network management to set up the environment is another important task.  
  
In some organizations, this task is controlled centrally. In these cases, a tester coordinates with the environment management for timely delivery of the test environment.

**implementing tests**

Testers implement test suites for all test levels of a test environment. These test levels may not necessarily fall under the direct control of the test group.  
  
For example, testers may have to implement unit tests that were designed by programmers. This would help them approach the software test from the perspective of the developer, potentially providing them with an improved level of understanding of the application and its development.

Testers also perform tasks such as

**reviewing tests**

The tester executes and logs the tests, evaluates the results, and documents the deviations from expected results as defects.  
  
For example, suppose the exit criteria that emphasize secure transactions are set for a payroll application. If these criteria are not completely met during the test process, the tester would identify the criteria mismatch from the actual results and then log them as defects.

**using testing tools**

The tester uses the test administration or management tools and test monitoring tools whenever required.  
  
As a tester, you would need to use specific defect-logging applications that enable you to record defects and assign them to developers. A working knowledge of the application helps you perform the test activities effectively.

**automating tests**

The tester automates tests or works with developers or other appropriate technology experts to do so.  
  
When you test the applications that have been built on recently updated technologies, you might need help from experts in that technology. This aids in understanding the system under test and also helps avoid logging invalid defects due to a lack of understanding of the updated technology.

**measuring performance**

The tester executes tests and measures the performance of components and systems if required.  
  
For example, tests implemented for research purposes may require you to measure and monitor the performance of the test components.

**performing peer review**

The tester reviews other testers' work, such as test specifications, defect reports, and test results.  
  
For example, a functionality tester may check if the defect report prepared by another functionality tester working on the same project is appropriate or requires modification.

Question

As a test leader, you must manage the test activities of a mail application. Which of these tasks should you perform?

**Options:**

1. Create test plans and test designs for specific functionality of the mail application
2. Review the defect reports of testers to ensure that they've detected all possible defects in the application
3. Identify which functionality of the application are tested and determine if it is according to the test plan
4. Specify the test condition that drafts are automatically saved after five minutes

Answer

***Option 1:****Incorrect. Test plans and designs are created by testers to provide an outline of the process involved in testing the application. Test leaders don't prepare these documents.*

***Option 2:****Incorrect. The tester performs a peer review of test specifications, defect reports, and test results of other testers.*

***Option 3:****Correct. The test leader compares the actual test progress with the planned progress and makes adjustments in the plan if there are deviations.*

***Option 4:****Incorrect. As part of creating test plans and test designs, the tester identifies test conditions and creates test designs, test cases, test procedure specifications, and test data.*

**Correct answer(s):**

3. Identify which functionality of the application are tested and determine if it is according to the test plan

Question

As a tester, you are required to test a spreadsheet application. Which of these tasks would typically be part of your tester role, as distinct from that of the test leader?

**Options:**

1. Specify a test condition as the ability to open and edit multiple instances of the spreadsheet
2. Prepare a summary report describing the working and failed functionality of the spreadsheet application
3. Document a defect found in the application – the Save functionality is not working
4. Assist other project activities, such as integration and planning to ensure seamless testing process
5. Assess the requirements of the users who frequently use spreadsheet applications

Answer

***Option 1:****Correct. As part of creating test plans and test designs, you have to identify the test conditions.*

***Option 2:****Incorrect.  Preparation of end-of-project summary reports is typically classified as a test leader function.*

***Option 3:****Correct. You need to execute and log the tests, evaluate the results, and document the deviations from expected results as defects.*

***Option 4:****Incorrect. The test leader provides testing perspective to other project activities, not the tester.*

***Option 5:****Correct. Testers need to assess user requirements to analyze test specifications.*

**Correct answer(s):**

1. Specify a test condition as the ability to open and edit multiple instances of the spreadsheet  
3. Document a defect found in the application – the Save functionality is not working  
5. Assess the requirements of the users who frequently use spreadsheet applications

**2. Summary**

The most common roles found on a test team are test leader and tester. Depending on the project requirement, an individual with the required knowledge and skills may perform one or both of these roles.  
  
Tasks assigned to the test leader include creating test policies and strategies, preparing a test schedule, measuring test progress, and preparing summary reports.  
  
Testers must have adequate skills in the application, technology, and testing domain to carry out their tasks. Tasks assigned to testers include creating test plans and test designs, setting up the test environment, implementing and reviewing tests.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#top)

Organizing a Software Test Team

Learning Objective

*After completing this topic, you should be able to*

* *evaluate the effectiveness of a test team*

**1. Exercise Overview**

**2. Test team for scheduling and budgeting**

**3. Test team for testing and fixing**

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#top)

Software Test Planning

Learning Objectives

*After completing this topic, you should be able to*

* *distinguish the different detail plans that are created from the master test plan, based on scenarios*
* *recognize the factors that influence the levels of a test plan*

**1. Components and kinds of test plans**

Software testing plays a major role in the software development process. Inadequate test planning can place an entire project's success at risk. The software test planning process includes organizing the test activities, identifying the resources, and assigning responsibilities to them. You also need to list the test procedures, strategy, schedule, and deliverables in the planning stage.   
  
You use the Test Plan document as a tool to plan the testing process for the software.

In the Software Test Plan document, you describe the objectives, scope, approach, and focus of a software testing process. A test plan also identifies the test environment, the design techniques, and identifies any risks involved in testing.  
  
Using the test plan, you can then communicate the test details to all the members of a project team. It helps them identify the test items, the testing tasks, and their responsibilities.

You plan a test in the early stages of a test project. However, as the project progresses, you can update your test plan according to the changes and requirements of the project. You can also write multiple test plans if the project is dynamic and involves continuous changes.

However, writing multiple test plans can lead to information overlapping and this information redundancy causes unexpected discrepancies in the plan statements. Therefore, it is better to write a single master test plan that addresses the common topics.

The IEEE 829 Standard for Software Test Documentation provides a template with standardized methods for performing tests and also helps you write, record, and track the tests throughout the life of a project.

A Standard Test Plan template contains 16 sections. These sections define the requirements of a project plan. Performing the test process according to the content in these sections enables you to meet the objectives with the available resources and within a stipulated time frame.

You can use the various sections provided by the IEEE 829 Standard Test Plan template to

**identify the test plan**

You can use the test plan identifier section to provide a unique number that identifies the test level, the software associated with the test plan, and the current version of the test plan.

**provide introduction**

In the introduction section, you can include the project scope description to identify the key features of a project. You can also specify the plan scope to address the different levels of testing.

**list test items**

You can list the items that need to be tested, such as source code and control data, in the test items section. The section also includes references to the design and requirement specifications.

**list features to be tested**

You can include a list of product features that might be required by the user in this section. For example, different users may require different performance and functionality levels. You can include these features in the test plan so that they can be tested.

**include features not to be tested**

In this section, you can include information about the features that do not require testing and provide reasons for their exclusion. For example, when a feature involves low risk and is not tested, it is listed in this section.

The other sections in the IEEE 829 Standard Test Plan template help you provide information such as the

**testing approach**

The testing approach or strategy is the most important section of a test plan. You include well-defined information regarding the testing approach and identify factors that may affect the success of that approach in this section.

**pass or fail criteria of test items**

You can add the pass/fail criteria of test items based on the number and severity of errors and their locations in the product under testing. However, these test criteria may differ from one organization to another.

**suspension and resumption criteria**

You can include information regarding the suspension and resumption criteria when you need to suspend or restart the testing process. For example, you can suspend the testing process when the tasks or the testing environments are incomplete.

**test deliverables**

In the test deliverables section, you can provide a list of all documents, tools, and components to be developed for the testing effort. Some test deliverables are test design specifications, custom tools, and simulators.

**test tasks**

You can identify a set of tasks required for planning and performing a test in the test tasks section. The test tasks section can also include details about the inter-task dependencies and the skills required to perform the test.

Other sections of the template include

**environmental needs**

You can include the testing environment requirements such as hardware, software, interfaces, and security access in the environmental needs section.

**responsibilities**

To specify the responsibilities of the people involved in the test and the tasks they perform, you use the responsibilities section. This section also includes information on the responsibility criteria of those who establish the test environment and manage the software configuration.

**staffing and training needs**

You choose the testing staff and train them based on the skill requirements for a particular testing process. In the staffing and training needs section, you can list the skill requirements and the training required for them on the testing methodology adapted.

**schedule**

You include key milestones of the test project plan such as document delivery dates and the available resources in the schedule section.

**risks and contingencies**

You can list project risks such as budget and resource availability, and contingency plans associated with these risks in the risks and contingencies section.

**approvals**

Using the approvals section, you collect the signatures of those who are responsible for final approval of the testing software. These individuals possess the authority to accept or reject the terms of a software document if they feel that the software requirements are not adequate.

You can easily remember the 16 sections of the IEEE 829 Standard Test Plan template by using the acronym 'SPACEDIRT'. The letters in the acronym stand for Scope, People, Approach, Criteria, Environmental needs, Deliverables, Test Plan identifier and Introduction, Risks, and Testing Tasks.

Question

You've created a test plan for testing a software product in your organization. There is a deadline for performing this test. When you execute the test, you identify that you need to stop a part of the testing activity due to a small problem and then resume it. Which of the components that you included in the test plan influences this test?

**Options:**

1. Approach
2. Suspension and resumption
3. Test items
4. Features not to be tested

Answer

***Option 1:****Incorrect. Before starting to test, you decide an approach to testing by referring to other documentation such as a test strategy. You then include it in the test plan.*

***Option 2:****Correct. During testing, you use the suspension and resumption criteria to determine stopping a part or all of the testing activity and resuming it.*

***Option 3:****Incorrect. You first decide the test items, such as source code or data, to be tested before testing it. Unless a test item has defects, it does not require you to stop or resume a test.*

***Option 4:****Incorrect. You usually determine the product features not to be tested based on the low risks they involve and the reason for their exclusion.*

**Correct answer(s):**

2. Suspension and resumption

A test plan is ideally a master test plan that accommodates all the revisions made to the plan as the test progresses. This helps you reduce redundant documentation. You can create the master plan during the early stages of a project and include the planned high-level test activities in it.

Suppose your organization has created test plans for a software product. The test plans ensure that the software must not contain critical defects and is capable of completing a task within the specified time. In addition, it should be compatible with the already developed software modules.   
  
You can choose acceptance testing to test these product features and ensure that it is completed on time.

To identify any critical defects and ensure that the software is compatible with the hardware, you perform system testing.   
  
You can check whether the software can operate with the modules already developed using integration testing. Checking the code or any component of the software can be performed using component testing. The test plans for all the Acceptance, System, Integration, and Component test levels are combined to create a master test plan.

You can create test plans at various levels and derive these plans from the master test plan. The various test levels derived are shown in the diagram.

Graphic

*Description of software test plans flow chart:  
  
A master test plan consists of Acceptance test plan, System test plan, Integration test plan, and Component test plans.  
  
Description ends.*

**Acceptance test plan**

You begin acceptance test planning in the early stages of the product life cycle after the high-level requirements are defined. Acceptance testing ensures that all requirements are met. For example, you can use acceptance testing to test the stress, timing, compliance, and capacity of software.

**System test plan**

System testing is comprehensive testing on the performance and reliability of the software. For example, suppose your company has outgrown its current payroll system. Management wants to develop a new system that facilitates new features. You have to test whether the system guides the employees with menu options and error messages. You can test this using system testing. A System test plan is based on the requirement and design specifications and should be in accordance with the master plan.

**Integration test plan**

You ensure that all the components of a system interact and function cohesively through integration testing. You can start the integration test planning when you are stabilizing the software design. In integration testing, you must test the critical components first so that errors in these components don't show up at a later stage in development resulting in costly delays.

**Component test plan**

You can test individual software components, such as the individual units of source code, through component testing. You can use component testing to detect errors early in the development cycle. This makes integration testing easier because all components are tested before they are integrated. For example, you can test units of code as separate components before integrating them.

Question

Match the types of software test-level plans with the appropriate scenarios.

**Options:**

1. Acceptance
2. System
3. Integration
4. Component

**Targets:**

1. Software must meet the needs of the users as specified in the requirements
2. You must determine the effectiveness and performance of a network
3. You must verify through testing that two software programs are able to communicate with each other
4. You must test the separate units of a software product

Answer

*You can test whether the software meets the user functional requirements through acceptance testing.*

*By using system testing, you can check the performance of the software. You can also determine the effectiveness of a network by system testing.*

*You can check the interoperability of software programs by performing integration testing.*

*You can check different units or components of a software product by component testing.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

**2. Factors influencing a test plan**

Test planning is the most prominent phase in any test project. A test plan lists all the tasks and key areas in a test along with the estimated test effort. You can use the test plan when you develop, implement, and maintain software.

To test a product efficiently, you must clearly understand the factors that can influence the test planning process and consequently, the test plans.

Graphic

*Description of the factors influencing the test planning process.  
  
The people, product, process, and approach factors are responsible for test planning.  
  
Description ends.*

**Product**

Product knowledge helps you perform efficient testing. Any risks that can cause the software product to fail should be understood at an early stage in development. Failure of software involves heavy cost and labor that could be avoided if diagnosed and resolved before or during development. For example, synchronization issues in the audio software of high-end digital music systems are very expensive and time consuming to rectify.

**Approach**

Different approaches are required for testing different types of software products. For example, software relating to an online banking system requires a more formalized level of testing than software relating to a basic website. The online banking system carries much higher risks and costs associated with failure relative to the basic website.

**Process**

Test planning is influenced by factors affecting the testing process such as the availability of testing tools. Using these tools can reduce the manual effort that would be required for executing a test. For example, in safety critical environments, such as the aviation industry, you can use debugging tools to reduce the cost and time required to complete a test.

**People**

One factor with a significant effect on the testing process is people. This factor can include the skills of the individuals on the test team and their relevant experience with similar software products and testing projects. For example, when testing a highly technical product to be used by clients with a technical background, the test team should be populated with testers that possess similar technical knowledge.

Question

Your client is currently using a product that has a similar interface design as a product that you are developing. Your product must be interoperable with your customer's current product. Therefore, you need to ensure that the timing, capability, capacity, and interoperability of your software is tested completely. You want to automate the testing of all program code so that you can complete the test within a relatively short time frame.  
  
Match the features of different test levels required for this scenario with the factors that influence them.

**Options:**

1. Knowledge about the interface
2. Performance capability requirements of the software
3. Availability of testing tools
4. Availability of expert code testers

**Targets:**

1. Product
2. Approach
3. Process
4. People

Answer

*When you test the user interface, including the login screen, you require thorough knowledge about the product and its interface because any failures may cause security issues. This involves system testing.*

*You should decide the approach to be used for testing the performance, capacity, and timing of the software based on its complexity and requirements. All requirements are tested through acceptance testing.*

*When automating testing, you require tools to minimize the manual labor and the cost associated with the testing process. Some test automation tools can be effectively used during integration testing.*

*Inspecting program code requires experts in the domain and therefore people factors influence this test significantly. Testing code involves component testing.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

**3. Summary**

A test plan documents the test planning process. You can develop a standardized test plan template using the IEEE 829 Standard for Software Test Documentation. The IEEE 829 template contains various sections to create virtually any type of software test plan for your organization.  
  
A master plan is created in order to avoid redundant documentation that would be covered in multiple test plans and addresses software test levels such as acceptance, system, component, and integration.  
  
You can perform effective and efficient testing using a properly documented test plan. Test planning and estimation is influenced by factors such as people, product, process, and approach.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#top)

Test Estimation, Test Strategies, Entry and Exit Criteria

Learning Objectives

*After completing this topic, you should be able to*

* *recognize different test strategies in a given scenario*
* *identify key factors that influence test effort*
* *recognize entry and exit criteria*

**1. Estimating tests using test strategies**

You should analyze a software test project in order to obtain an accurate estimate. This will enable you to check if the available resources such as the test environment, funding, and the people are adequate for the project. You can then use the test estimate data to assess risks and ensure timely delivery of a project.  
  
Before delivering a product, you need to also ensure that you successfully complete the test plan and that it meets all the necessary exit criteria with no outstanding critical errors.  
  
Successful completion of a test is based on the approach you adopt for testing. You can decide a testing approach according to the risks, cost and time requirements of a project.

To estimate the overall cost for a project or its duration, you can use two approaches:

**metric-based**

You can use the metric-based approach when you rely more on the test data for estimation. For example, you can analyze the number of test cases to determine the testing cost. Similarly, you can use other data, such as written and executed test cases, the time required to run and execute them, and the defects found, to calculate the estimate.

**expert-based**

You can use the expert-based approach when you want to use an expert's experience for test estimation. These experts may be technical, analytical, or business experts. With their guidance, you can determine the test scope and the risks involved.  
  
You can then divide the testing process into several tasks based on the expert's data and calculate the cost, time, and the resources required for each task separately. By summing up the data collected for all the tasks, you can estimate the cost of the whole testing project.

These approaches, when combined, can estimate a test more accurately than when handled separately. For example, based on the expertise of the testing team in your organization, you can breakdown a project into several tasks to estimate the testing effort needed for each task.

Then you identify similar metrics from past projects such as the number of tests run and the number of defects found per day. You then apply these parameters to calculate the test duration and test effort for all key tasks in your project. Finally, you estimate individual tasks and obtain a project level estimate.

Question

Match the features of each approach to the respective approaches. You can use each target more than once.

**Options:**

1. Calculate test estimate based on the number of defects
2. Receive estimates from technical architects
3. Validate cost on the number of executed test cases
4. Determine the scope and risks involved in a project

**Targets:**

1. Metric-based
2. Expert-based

Answer

*The metric-based approach helps you estimate the testing effort of a project by referring to similar previous projects. Based on the parameters, such as the number of defects found by a tester per day and the cost involved for the number of test cases in these projects, you derive a final estimate for your project.*

*Using the expert-based approach, you gain knowledge from experts, such as technical architects, to test a software product. These experienced experts guide you in determining the scope and risks of a testing project.*

**Correct answer(s):**

Target 1 = Option A, Option C

Target 2 = Option B, Option D

Testing is a complex activity, and the testing effort is influenced by a variety of factors. When you create test plans or estimate the test effort, you must concentrate on these factors to complete the test successfully.

The key factors that influence the testing effort are

**product**

You can use project documentation to obtain information about the software product. Adequate information, such as usability and performance requirements of the software, helps you to plan effective and efficient tests. For example, managing timing constraints for an avionics mission control computer system requires you to have intricate knowledge about the product's requirements.

**process**

The cost of a test project is directly proportional to the size and life cycle of the test project. If the test process is mature enough to enable effective management of changes, you can reduce test execution cost by avoiding rework costs. You can also reduce test execution costs by automating tests through test tools. This also helps reduce the time required to complete testing.

**people**

A good test team with relevant experience and skills is more capable of performing efficient testing. The team must be reliable in order to commit to timely delivery of quality projects. The stability and geographical distribution of the team also plays an important role in influencing the test effort. For example, on a team that changes often, the experience and knowledge levels may differ and that can adversely affect testing efficiency.

**test results**

Test results are also important to the total test   
effort. If you deliver high quality software and quickly fix errors, you can prevent delays in test execution and ultimately product delivery. For example, once a defect is documented, if it is addressed promptly it will not likely have to go through multiple iterations of fixes and retesting, making it much easier to meet the initial test estimate.

Question

Match the examples with the respective factors that influence a test effort.

**Options:**

1. You need to test software related to spacecraft
2. You must use automated tools for testing
3. You need to consider the geographical location of the testing team when you plan a test
4. You must deliver high quality products

**Targets:**

1. Product
2. Process
3. People
4. Result

Answer

*Software systems related to spacecraft are highly technical and you require adequate knowledge about the product to test it.*

*You use tools to speed up a testing process and reduce the effort of manual labor in the process.*

*People at the same location can communicate their ideas more effectively than a team dispersed across locations. This results in efficient testing.*

*When you deliver high quality products, you get effective test results, thereby reducing the overall test effort.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

Choosing an appropriate test approach or strategy plays an important role in the success of a testing project. A test strategy defines the test levels and their corresponding activities within the project. Depending on the project, you can use different testing strategies for each level.

When you implement one of these strategies on a project, you effectively define an approach for the project tests. With this strategy-enabled approach, you understand how to perform a test, identifying the requirements for the tests from beginning to end. The strategy you apply also helps you to identify the risk analysis to be carried out, the design techniques to be applied, and the exit criteria to be applied.

The various software testing strategies include

**analytical**

You use an analytical strategy when you analyze the requirements of a project and the associated risks involved. You plan, design, and estimate tests based on the parameters derived from this analysis. For example, you analyze the requirements specification of a project and design a test process based on these requirements. You then estimate the cost and effort required to execute the test process.

**model-based**

In model-based testing strategy, you derive test cases from a model that describes some functional aspects of the product under test. Model-based testing presents an excellent candidate for automation. For example, you can run tests automatically instead of hand-crafting test cases to meet the short release cycles in many markets today.

**methodical**

For a methodical test strategy, you use a planned, systematic approach through which you gather information from major areas of testing and develop a checklist based on this information. You then design, implement, and execute tests based on this checklist.

**standard-compliant**

You can adopt a standardized strategy for testing. These standards often depend upon an externally developed approach to testing. For example, you can follow the IEEE 829 standard or similar standards for developing a template testing strategy.

**dynamic**

By employing a dynamic strategy, you can concentrate on finding the maximum number of defects possible during test execution and resolve them. For example, exploratory testing is a dynamic strategy where you can control the design of the tests when they are performed. You can use the knowledge obtained from this to design more efficient tests. You can apply this form of testing at any stage in the development process.

**consultative**

You can use a consultative strategy when you need to consult with the developers or users for guidance on the product. These resources are normally from outside the test team. They can guide you on product technologies and other items in the testing process.

**regression-averse**

Regression occurs when a software upgrade identifies errors that were previously not observed. In the regression-averse strategy, you re-run previous tests and use automated testing tools to run all regression test cases and report failures. For example, you modify program code and run the existing tests to check for any variation in performance of the modified code.

You can choose between test strategies that are preventive or reactive. For example, analytical test strategies identify defects prior to test execution and so they are preventive. Dynamic test strategies locate defects during the test execution process and enable you to solve them reactively.

You may also combine any of these strategies to suit a particular test requirement. For example, a tester wants to explore, write, and execute test scripts simultaneously. You then need to consider risks that may cause test cases to fail so that you can avoid or correct them. In this case, you can blend the analytical, dynamic, and regression-averse strategies to satisfy all conditions.

Blending strategies to enhance the success of a testing effort depends on many important factors. Some of these factors are

**risks**

Risk-management is an important factor to consider in any project. You need to identify, analyze, track, and communicate risks in software intensive programs as they can affect the potential for success of a project. For example, if you identify that a product will not meet the performance or functionality requirements within the defined schedule, it is a risk that you need to focus on resolving.

**skills**

To execute the chosen strategy, you must consider the skills that a test team has and the lack of skills as well. When a team lacks skill and time is short, the best strategy to choose is one based on standards rather than creating a new approach.

**objectives**

To perform successful testing, you must ensure that your test meets all objectives of the testing strategies used. If your objective is to find the maximum errors possible within a stipulated time and effort, then a dynamic strategy is appropriate.

**regulations**

In addition to meeting objectives, your test should also satisfy any regulations that may exist. You may need to develop a methodical test strategy to ensure that your test also meets all the necessary regulatory requirements.

**product**

Some software products, such as those used in weapon systems, have stringent requirements. You need to analyze whether the software meets these requirements. Therefore, applying an analytical strategy is appropriate in this case.

**business**

Business considerations are essential as is business continuity. If you can use an existing system as a model for new development, a model-based strategy is a good candidate.

Question

You have a short time frame available for testing a software project in your organization. To identify the areas on which the test is to be performed, your organization follows an industry-standard for software quality. Which testing strategies will you apply for testing?

**Options:**

1. Analytical
2. Model-based
3. Methodical
4. Consultative

Answer

***Option 1:****Correct. Within a short duration, you may not be able to perform thorough testing. Using the analytical strategy, you can analyze the risks involved in the project and concentrate on the areas of highest risk.*

***Option 2:****Incorrect. A model-based test strategy includes the creation or selection of some formal or informal model that closely matches the software project under test. This is not an industry standard strategy.*

***Option 3:****Correct. Methodical test strategies involve a systematic approach that enables you to use a standard outline and methodically design and execute your test according to this standard.*

***Option 4:****Incorrect. A consultative strategy allows you to consult the users or developers of a software product during testing. This strategy does not use any formal or standardized approach.*

**Correct answer(s):**

1. Analytical  
3. Methodical

**2. Entry and exit criteria in software testing**

Entry criteria are used to establish the conditions and timing for a given test activity to commence.  This may include when to start test design, when to start a new level of testing or when to start test execution. Without such criteria, certain test activities may begin at an inappropriate time, using valuable resources, which in turn wastes time and money on a project.  
  
For example, suppose that the test team started testing code which was ready, but the correct data for the testing execution was not at a ready state.  The test team could waste significant valuable time against the project and may have to go back and re-test once the data is ready.

Usual entry criteria include:

**test environment availability**

Test environment availability indicates that any system that will be used for testing purposes is accessible to those that are doing the testing.  For example, if the system is maintained remotely, then the testers must be able to remotely connect to the system via Remote Desktop (RDP) or similar remote access protocol.

**test tools installed and available in test environment**

In addition to ensuring that the test environment is available, the environment must also include the test tools installed and configured according to the specifications outlined in the test plan.  For example, if the software test plan calls for the use of a particular test tool, that tool must be installed and configured explicitly for testing purposes.

**code is available for testing**

The code must be available to testers in order to perform the required tests according to the test plan.  For example, to perform static testing, testers will commonly use a compile environment to ensure syntax validity.  Without the code, this test would be impossible.

**test data available and ready to use**

This type of entry criteria ensures that the test data indicted by the test plan has been entered in the required environment and is ready to use for testing.  For example, if the software to be tested interfaces with a relational database, the data required in order to run the tests must already exist in the database (such as user roles for login example).

**test design activity is complete**

In order to carry out testing effectively with no wasted effort, it is essential that the test conditions (coverage items) for a test item, the detailed approach and associated high level test cases are fully documented and complete.  For example, if the test condition calls for a certain approach in testing, and a different approach was employed instead, all testing performed using the erroneous approach would likely amount to wasted effort.

Even though you may use all the indicated and appropriate strategies, inadequate requirements or resources may force you to stop testing a project. To prevent this, you must ensure that your test meets the conditions of exit criteria for successful completion.   
  
Exit criteria identify any inadequacies and incomplete testing tasks during project execution. For example, an exit criteria condition may require you to check if any severe defects are left outstanding before testing can be formally considered complete.

There are various types of exit criteria. You may classify them as

**defects**

When testing a project, the exit criteria for defects require you to meet certain conditions, for example, that high priority errors must be rectified and retested. This helps you ensure that no critical outstanding errors are present in a project before releasing it.

**coverage**

You can improve the quality of a project by maximizing the test coverage. Test coverage covers all the test levels. Exit criteria for test coverage include conditions. For example, they may include a condition that says 85% of requirements coverage is to be achieved in order for a test to be considered complete.

**quality**

Exit criteria provide an effective method for managing the quality of a project. You can use exit criteria to ensure that you have successfully completed a test and the outcomes are of acceptable quality. For example, the criteria for quality may require you to execute maximum test cases with an item pass rate of not less than 80%.

**budget**

You can stop testing when the budget for the test is exhausted. Exit criteria determine when to stop according to available funding. Using these criteria, you can also rectify your test plan to complete the test within the specified budget.

**risk**

Using exit criteria, you can ensure that all potential areas of risk such as technical and management risks are completely tested. Exit criteria determine the product features at risk which allows you to leave the low risk areas untested if necessary.

Different exit criteria relate to different levels of testing. For example, checking whether a project is code complete and there are no missing features or media elements is considered component testing. Testing whether a product runs on all the product's supported hardware and software configurations is considered system testing.

You can check whether the outcomes are of acceptable quality by using acceptance testing. Integration testing requires the software product to be compatible with and installable on all other related systems. If a product is approved and signed off by product marketing, the performance of the product is confirmed and approved by functional testing.

Question

You are testing a financial application. You need to meet certain exit criteria for the successful completion of the test such that the test should conform to the budget and 85% requirements coverage has to be achieved during testing. Which types of exit criteria do you need to meet these specifications?

**Options:**

1. Coverage
2. Budget
3. Defects
4. Tests

Answer

***Option 1:****Correct. You can improve the quality of a project by maximizing the test coverage. In this example, the exit criteria for test coverage included the condition that 85% of requirements coverage is to be achieved in order for a test to be considered complete.*

***Option 2:****Correct. Completing a testing project within the limits of a planned budget requires accurate planning of the cost effort for the project.*

***Option 3:****Incorrect. The specifications mentioned do not have any criteria for the defects.*

***Option 4:****Incorrect. The criteria are for the results of testing and not for the test itself. The tests should be successful with 85% requirements coverage.*

**Correct answer(s):**

1. Coverage  
2. Budget

**3. Summary**

Test estimation defines the scope, time, resources, and cost required for testing. There are two approaches for testing – expert-based and metric-based. You combine these two approaches to arrive at a reasonably accurate estimate for the test. Software testing is influenced by factors such as product, process, people, and test results. You must use these factors properly to reduce the test effort. In software tests, you can apply test strategies that you can deliver through suitable approaches.  
  
You may classify test strategies as analytical, model-based, methodical, standard-compliant, dynamic, consultative, or regression-averse. You can blend these strategies depending on a particular scenario. The factors that help you combine these strategies relate to risks, skills, objectives, regulations, product, and business.  
  
You must ensure that the strategies and approaches you adopt meet the entry and exit criteria of a testing project. You can define entry and exit criteria for all testing activities. These criteria relate to the different test levels – component, integration, system, and function testing.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#top)

Software Test Progress Monitoring and Control

Learning Objectives

*After completing this topic, you should be able to*

* *interpret test metrics*
* *evaluate test summary reports and select test control actions*

**1. Interpreting test metrics**

During testing, a testing team's objective is to check whether a product meets the specified requirements. To ensure that the team successfully achieves its objectives, you need to monitor all testing activities. For effective monitoring, you collect data related to each testing activity and use it to assess whether the activities are progressing as planned. This data is known as *Metrics*.

Typically, you collect metrics data during and at the end of each test level. This data helps you easily verify whether the testing activities adhere to the allocated schedule and budget. It also helps you verify the relevance and effectiveness of the test objectives and the testing activities.

Some of the metrics you collect include

**percentage of work done**

To determine the work percentage, you calculate the number of test cases the testers prepared and the time they took to prepare the test cases. You also calculate the percentage of time and effort the testers spent preparing the test environment. These calculations help you determine whether testers are working to meet their targets and objectives.

**performance of test cases**

You can also use test cases as metrics to monitor the progress of testing. To do this, you determine if each test case was performed, and you further identify the performed test cases that ran successfully. By monitoring test cases, you can validate the time spent by the testers in preparing the test cases.

**defect information**

When collecting metrics, you also gather information about defects the testers identify. For example, you determine the *defect density*, which is the ratio of the number of defects testers identify to the size of the component or the system they are testing.   
  
Other defect-related data you collect include the number of defects that were fixed and the number of times a fixed component or system had to be retested.

**extent of testing**

The extent to which testers test a component or system is also an important metric. For example, you check whether testers tested each code segment thoroughly or if they verified the potential risks of a component or system alone.   
  
The extent of testing varies with the test level. For example, testers test code segments during component testing and verify system requirements during system testing. By monitoring the extent of testing, you can ensure that each test level progresses as planned.

**confidence level of testers**

You can also use the confidence level of testers in the quality of the product being tested as a metric. If the testers are too confident about the quality, they may not test the product thoroughly and so may overlook serious defects. On the other hand, if the testers are not confident of the quality, they may put the product through unnecessary tests, which can only prolong the testing process. You should try to ensure that the testers adopt a balanced and bias-free approach to testing.

**deliverable dates and testing costs**

You collect data related to deliverable dates and testing costs as metrics to determine if each test level can be completed on time and within budget. For example, if you find that a testing activity is overshooting the allocated budget, you can decide whether to stop the activity, modify its scope, or perform it according to the existing plan. You base your decision on the impact the testing activity can have on other testing activities and the overall progress of testing.

Question

You are monitoring the testing of a software product. During the monitoring process, you record how many of the planned test cases have been prepared, and and how much of the test environment is configured. Additionally, you are recording the number of defects and the testing effort required to find these defects. Which types of metrics are you collecting?

**Options:**

1. Percentage of work done
2. Performance of test cases
3. Extent of testing
4. Confidence level of testers

Answer

***Option 1:****Correct. Keeping track of the number of planned test cases completed, and the extent to which the test environment has been built,  will provide you with a percentage work done metric.*

***Option 2:****Correct. To verify the success rate of test cases, you monitor their performance. Using this metric, you can validate the time spent by testers to prepare the test cases.*

***Option 3:****Incorrect. You monitor the extent of testing to ensure that each test level progresses as planned. For example, you check whether testers test the interactions between software components during integration testing. You don't use this type of metric to track whether or not testers perform unnecessary quality checks.*

***Option 4:****Correct. Often, if testers are not confident about the quality of a product, they subject it to unnecessary tests. This effort may not serve any useful purpose and only prolong the testing process. So the ratio of testing effort to defects found will provide an indication of the testers subjective confidence level in the product.*

**Correct answer(s):**

1. Percentage of work done  
2. Performance of test cases  
4. Confidence level of testers

You can use various tools to record and display test metrics. For example, you can record the data in a textual format in an Institute of Electrical and Electronics Engineers (IEEE) 829 test log template or in Microsoft Office Excel worksheets. You can also represent the data in a graphical format using bar graphs.

If you only want to maintain a simple record of the tests the testers perform and the defects they identify, you use an Institute of Electrical and Electronics Engineers (IEEE) 829 test log template. This template consists of three standard sections: Test log identifier, Description, and Activity and event entries.

Graphic

*In the example, the different sections of the standard IEEE 829 test log template are displayed.*

In a test log you create using the template, you first add a unique number to identify the log in the Test log identifier section. Then, in the Description section, you describe the software components being tested and the test environment.

The last item you add to the log is information about testing activities and events. For example, you mention the testing process, the outcome of the testing process, defects identified, and a description and location of the defect report.

If you want a more detailed record of the metrics, you can create test-summary sheets using Excel. For example, you can use a summary sheet that contains an overall representation of the test metrics, such as bug ID and planned and actual dates, actual effort, and the test duration, in various columns.

Graphic

*Description of the EasyNomadTravel System Test Summary Cycle One worksheet:  
The worksheet consists of 13 rows named Test ID, Test Case, Status, System Config, Bug ID, Bug RPN, Run By, Plan Date, Act Date, Plan Effort, Actual Effort, Test Duration, and Comment.   
Description ends.*

In the Test ID and the Test Case rows of the summary sheet, you store the ID and the name of each test case, respectively. For example, to enter the data for a functionality test, you assign the ID as 1 and the Test Case name as Functionality.

Under each main test case, you also add the names of its specific sub tests. For example, under Functionality, you can include data for tests conducted on menu items of a software product and its formatting and printing capabilities. You then summarize the data related to each test case and its sub tests against the Summary entry.

In the Run By row of the summary sheet, you specify the initials of the person who conducted the test. Similarly, in the Test Duration row, you record the duration of each test and you use the Plan Effort and Actual Effort rows to keep track of planned and actual testing efforts.  
  
Other useful information on the summary sheet includes

**Status and System Config**

You store the status of each test case in the Status row and the code assigned to each type of testing environment in the System Config row. For example, you specify a status of Pass if a test case runs successfully. Similarly, you specify a status of Warn for a test case that results in a minor failure, the status of Fail for a test case that fails completely, and the status of Skip for a test case that the testers don't perform.  
  
The code you type in the System Config row is defined in the test plan document.

**Bug ID and Bug RPN**

In the Bug ID and Bug RPN rows, you record the ID and risk priority number (RPN) of each bug, respectively. You use the defect-tracking database to determine the ID of a bug. You use values 1 through 25 for the RPNs. These values are in descending order of risk level. For example, you assign the RPN value of 1 to the most risky bug.

**Comment**

In the Comment row, you store specific information related to a test case or sub test. For example, if the testers didn't conduct the Printing capabilities sub test, you type the text "Not conducted" in the Comment row.

You can also track the progress of testing activities and test levels based on the number of defects found and closed. To depict the defect data for a testing activity or test level, a graphical format is most suitable. A graph helps you check the overall status of testing easily and quickly.

Graphic

*Description of the Integration Test Metrics line graph:  
  
The example line graph depicts the number of defects opened and closed during integration testing. As the end date of integration testing nears, the gap between the lines representing the total number of defects opened and the total number of defects closed diminishes.  
  
Description ends.*

By using a graph, you can also show the planned and actual start and end dates for a testing activity or test level. Additionally, you can plot the number of defects the testers are expected to find and the number of defects they actually found. The graph also helps you monitor the difference between the number of defects found and closed on specific days.

The graph in the example indicates that the difference between the number of defects open and closed decreases on a holiday. This is because testing activities often decrease or stop during and immediately before holidays.

Similarly, stringent testing on Fridays results in more defects being identified on that day. So the difference between the number of open and closed defects increases on Fridays. Also, as testers verify fixes and close defects on Mondays and Tuesday, the difference between open and closed defects drastically decreases on those days.

When the gap between the number of defects open and the number of defects closed reduces to an acceptable level, the product is fit to be released.

Note

*You shouldn't depend solely on a defect-data graph because defects can easily be closed without being fixed.*

An alternative way to represent defects in a graph is by depicting the*failure-rate* or defect-density. You use this type of graph over a period to track the progress of testing. You declare a product fit for release when the failure rate or defect density falls below a pre-specified value.

Graphic

*Description of Failure-rate graph:  
  
The graphs depicts the Number of failures detected every hour during component testing.   
  
Description ends.*

A failure-rate or a defect-density graph proves useful if you want to create a highly reliable product. You calculate failure rates of a product or a component in terms of a unit of measurement. For example, you calculate the number of times a product fails per hour. You can also use the number of times the product or component is run as a unit of measurement.

**2. Evaluating reports and controlling tests**

You need to share the data and metrics you collect while monitoring a test level with other members of your team. To share the data and metrics in a format that is easy to interpret, you prepare a test summary report.

Your team members can then use the summary report to quickly check what events occurred during a testing activity or test level. They can also use the report to quickly learn about existing defects in a software product or component, the risks involved, the confidence level of testers in the product or component being tested, and the feasibility of repeatedly testing a product or component.

You can use the IEEE 829 Standard: Test Summary Report Template to create a test summary report. This template consists of eight sections.

You use the Test summary report identifier section to assign a numeric identifier to a test summary report. This identifier helps configuration managers easily identify and track the report.

In the Summary section, you include a summary of all testing activities that were part of the test level. Your team members can then use this section to quickly gather information about test-design specifications, test cases, test procedures, and test environments. This section also enables you to identify defect details, version numbers and descriptions of the products and components that were tested.

The Variances section refers to the variations between the actual testing strategies, specifications, and procedures and guidelines mentioned in the test plan. A testing manager can use this section to learn how to improve and streamline future testing activities and ensure that the activities don't deviate from the plans.

The other five sections of the template include

**Comprehensive assessment**

In the Comprehensive assessment section, you specify the extent to which the testers completed the testing activities. In this section, you also explain whether or not the activities conformed to the test-plan guidelines. Additionally, you include a detailed reference to the steps the testers took to ensure successful and efficient testing.  
  
Using this section, you can determine the extent to which a product or component meets the exit criteria at the end of each test level.

**Summary of results**

The Summary of results section provides a summary of the output and results of the testing process. This section also includes a summary of all types of defects and the defect-resolution process.

**Evaluation**

Using the Evaluation section, you evaluate each tested component or product against the specified requirements and metrics. For example, during integration testing, you evaluate the interaction between software components and record the expected and unexpected behavior of the components.  
  
In the Evaluation section, you also specify the scenarios under which the tested component or product may fail. This information is based on the stability and reliability the component or product exhibited during testing.

**Summary of activities**

In the Summary of activities section, you include a summary of the major testing activities and events. You also document the total resources, time, and budget you allocated to testing. For example, you document the total number of testers and computers you employed and the duration for which you assigned them to the testing activities.  
  
A testing manager can use the information in this section to estimate the requirements for future testing activities.

**Approvals**

You record details about the parties responsible for approving the test summary report in the Approvals section. You also obtain their signatures in this section. The signatures certify that the parties responsible for approval understand and accept the test results.

Question

Your team needs to perform system testing on a shopping portal. Integration testing on the portal was performed by another team, and the team has sent you a test summary report for reference. Which section in the report helps you verify whether the exit criteria for integration testing were met?

**Options:**

1. Evaluation
2. Approvals
3. Summary of activities
4. Comprehensive assessment

Answer

***Option 1:****Incorrect. The scenarios under which a tested component or product can fail are specified in the Evaluation section. This section also indicates whether the tested component or product meets the pre-specified client and product requirements and metrics.*

***Option 2:****Incorrect. The Approvals section contains information about the people responsible for approving the test summary report.*

***Option 3:****Incorrect. The Summary of activities section contains a summary of the major testing activities and events. A testing manager can use the information in this section to estimate the requirements for future testing activities.*

***Option 4:****Correct. You can use the Comprehensive assessment section to determine whether the portal meets the exit criteria. This section also helps you assess the comprehensiveness of the testing activities and determine whether the activities conformed to the test-plan guidelines or not.*

**Correct answer(s):**

4. Comprehensive assessment

Each testing activity can be delayed by risks. For example, testers might be unable to complete system testing on time because they don't have the administrative rights to access the product. In such cases, you can avoid or minimize the impact of risks on the testing activities by taking measures to control the tests.

Test-control decisions are based on the test metrics and information in the test summary report. For example, if the test summary report indicates a deviation from planned testing activities and the planned testing schedule, you take control measures to minimize the deviation.

As a testing manager, the test-control activities you need to perform include

**analyzing test-monitoring data**

You analyze the test-monitoring data and make test-control decisions based on the data. For example, if you find that a higher percentage of test cases fail, you may suggest more stringent guidelines for preparing test cases. Similarly, if you find that the testers are ignoring potential risks during testing, you can ask them to test the product against those risks.

**changing test schedule and priority**

To accommodate delays, you change the test schedule and the priority of the testing activities. For example, if testers don't receive a crucial component for testing on the scheduled date, you change the priority of the testing activities and allow testers to test components available to them. Similarly, you can reschedule testing activities if the test environment is not made available to testers on the scheduled date.

**setting acceptance criteria**

You can minimize or eliminate defects in a fixed component or product by setting the acceptance criteria for retesting the component or product. For example, you can allow testers to accept the component or product for retesting only if the developers first thoroughly test it and declare it defect free. This type of testing, which is conducted by developers, is known as confirmation testing.

**reviewing risks**

You can control tests by periodically reviewing risks. For example, during the course of a testing activity or a test level, the risks to a product may change. Existing risks may be eliminated or become less severe, or you may discover new risks. You can then review these risks and change ratings of the identified risks. This also helps you avoid unnecessary tests and allows you to perform efficient testing in less time.

**adjusting scope of testing**

If new features are added or changes are made to a product during a late stage of development, you need to adjust the scope of testing accordingly. For example, if a client requests that new features be added to a software application that has already been integrated, you include additional tests to check the new features.

There are also a few test activities that the project manager rather than the tester or testing manager is directly responsible for. To help project managers handle these activities efficiently, you can share your test-control experience with them and make suggestions.

For example, if project managers want to quickly deliver a prototype of the product to the client, you can suggest that they reduce testing time and effort by including only the crucial components in the prototype.  
  
Similarly, if the project mangers want a highly reliable product, you can suggest that they put the product through all necessary tests and release it only when it meets the exit criteria. You can also recommend that they test the product in the production environment to detect and eliminate all possible defects.

Question

You're the testing manager for a team that's testing an order-processing system for a chain of department stores. You anticipate a delay in the delivery of various system components to your team. Which test-control measure is most appropriate in this case?

**Options:**

1. Change the priority of each testing activity
2. Review the risks to the testing activities
3. Set acceptance criteria for each component
4. Release the product only after it meets all exit criteria

Answer

***Option 1:****Correct. To accommodate delays, you can change the priority of the testing activities and allow the testers to test the components that are already available to them first.*

***Option 2:****Incorrect. The most obvious risk in this case is the testing schedule and not the testing activities. The risk reviews only help you change the ratings you assign to the identified risks.*

***Option 3:****Incorrect. You set acceptance criteria for components that are being retested. The acceptance criteria help ensure that developers minimize defects in or remove all defects from the components they send for retesting.*

***Option 4:****Incorrect. It's the responsibility of project managers to change the release date of a product. You can only make recommendations to them about when they should release the product.*

**Correct answer(s):**

1. Change the priority of each testing activity

**3. Summary**

While monitoring a testing activity or test level, you collect and use metrics to measure the progress of the activity or level. You can record the metrics using test summary sheets or graphs. The summary sheets help you keep track of data related to test cases. You can use graphs to track data related to the number of defects open and closed, the failure rate of a component or product, or the defect density.  
  
To help your team members easily understand and interpret monitoring data and test metrics, you prepare a test summary report. You can use the IEEE 829 Standard Summary Report template to create the report.  
  
To ensure that tests progress as planned, you perform various test-control activities. Some test-control activities, such as changing test schedules and priority, are your direct responsibility. Others, such as delaying the release date of a product, are not. However, you can make recommendations on how to best handle activities that are not your responsibility.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#top)

Test Reporting and Summarization

Learning Objectives

*After completing this topic, you should be able to*

* *prepare a test plan*
* *analyze a test case summary*

**1. Exercise Overview**

**2. Determining a test plan and approach**

**3. Analyzing a test summary**

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a06_it_enus/output/html/course_transcript.html#top)

© 2016 Skillsoft Ireland Limited