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

**Course Transcript**

Testing Throughout the Software Life Cycle

**Software Development Models and Testing**

| [1. Software Development Models](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#t2) |

| [2. Component and Integration Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#t5) |

| [3. System and Acceptance Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#t8) |

| [4. Testing to Reveal Defects](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#t11) |

**Software Test Types**

| [1. Functional and Non-functional Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#t15) |

| [2. Structural and Changed-based Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#t17) |

| [3. Maintenance Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#t21) |

| [4. Models of Software Testing and Development](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#t24) |

Software Development Models

Learning Objectives

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

* *recognize how software testing relates to the development life cycle*
* *identify software test levels*

**1. Software testing and development models**

Depending on the requirements of a software-development project, you use a specific development strategy. For example, if you want to quickly design and develop software, you use a fast-track approach. However, if your main aim is to develop a high quality product, you spend extra time on the design and development process.   
  
Your development strategy depends on the availability of time and resources, the allocated budget, and the scope of the project.

Each development strategy is known as a   
*Software development model*. Each model defines the order in which you should perform project-related activities. These activities include identifying the client's requirements, creating a product design, developing the required product, and testing the product for defects. Testing is an important part of each model.

Software development models are of two types:

* sequential
* iterative

In a sequential development model, the project-related activities are distinct from each other. Only after you complete one activity can you initiate the next.  
  
The simplest sequential development model is the waterfall model. The waterfall model comprises six activities.

Graphic

*Description of the waterfall model flow chart:  
When using the waterfall model, you first identify client requirements. Second, you identify system requirements. Third, you create an overall design, and fourth, you create a detailed design. Then you develop the product, and finally, you test the product.  
Description ends.*

**Identify client requirements**

To develop a satisfactory product, the development team starts by identifying the client's requirements. During this activity, the team determines the client's expectations of the product. Then the team records these requirements in a requirement specification document.

**Identify system requirements**

After identifying the client's requirements, the team identifies system requirements or the requirements that the product should satisfy. This activity helps them identify the features and functions the software product needs to possess to meet the client's needs. The output of this activity is the functional specification document.

**Create an overall design**

Based on the identified requirements, the team creates an overall design of the software product. This overall design outlines the external features of the product. This design also describes the internal components of the product, such as modules and classes, and their interrelationships. The team saves the overall design in the technical specification document.

**Create a detailed design**

While creating a detailed design, the team defines the functions of each of the identified components and decides on the method to create the components. To store the detailed design, it creates the program specification document.

**Develop the product**

Using the detailed design as a blueprint, the team writes code for each component of the product and, finally, integrates the components to create the product.

**Test the product**

After developing the product, you test it to ensure that it meets the client and product requirements as well as the requirements of all types of end users.

The main drawback of the waterfall model is that it incorporates testing toward the end of the development life cycle. Because the code for the product is too complex at this stage, fixing bugs or defects in the product can be time consuming. Also, while fixing the bugs, you can introduce new bugs in the product.

To overcome the drawbacks of the waterfall model, you can divide the development of the product into various stages and perform a testing activity at the end of each stage. This division of development and testing activities can help you identify bugs in their nascent stage, when you can easily fix them.

The V-model is a sequential model that improves upon the waterfall model. In the V-model, the development team first identifies client and system requirements and creates an overall and detailed design. You and other members of the testing team review the documents generated at the end of these processes. You simultaneously plan for the testing activity.  
  
As the product is progressively developed, you and the other testers test it. However, you divide the testing into four test levels.

Graphic

*Description of the V-model flow chart:  
In the V-model, you first identify client requirements and plan for acceptance testing. Next, you identify system requirements and plan for system testing. The third step in the V-model is to create an overall design and plan for integrating testing. And the fourth step is to create a detailed design and plan for component testing. After the fifth step, you move on to developing the product. Finally, you progressively conduct component testing, integration testing, system testing, and acceptance testing.  
Description ends.*

**Component testing**

During component testing, you test each component of your product for defects. Also, you fix any defects immediately.  
  
You use the detailed design of the product as the basis for creating a component-testing plan.

**Integration testing**

Integration testing is used to test the interrelationships between the components of the product. During integration testing, you determine whether the components interact correctly with each other and with computer hardware and other software.   
  
To create an integration-testing plan, you refer to the overall design of the product.

**System testing**

Before you begin system testing, developers integrate all the components and build a functional software product. It exhibits the desired features and functions and performs without any evidence that there are defects present.  
  
You create a system-testing plan based on the system requirements.

**Acceptance testing**

Acceptance testing is conducted by the representatives of the client. The representatives may either test the product at your site or test it after you've delivered it to them. Their main objective during acceptance testing is to determine whether the product meets all their requirements.   
  
You create an acceptance-testing plan after the client's requirements have been identified.

In the V-model, you test the product at each stage of development. You release the outcome of each stage to the next stage only when all identified defects have been fixed.  
  
However, the V-model also has a drawback. In this model, you verify a product against client requirements only toward the end of the development process. Fixing bugs and adding missing features and functions to the product at this stage can be difficult, expensive, and time consuming.

The iterative development model eliminates the drawback of the V-model. In the iterative model, developers build a product using a series of iterative steps. Each step consists of four tasks – Identify requirements, Create a design, Create code, and Test code. You, as the tester, are part of the development process. Because you are able to test the product while it is being developed, you can identify bugs easily and accurately.

Graphic

*Description of the iterative model flow chart:  
In the flow chart, four rectangular boxes represent the four tasks – Identify requirements, Create a design, Create code, and Test code. Connected to the test code box is a decision box labeled "Have all requirements been met?". If the answer to this question is "Yes", you end the development process. Otherwise, you move back to the first task and repeat the entire process.  
Description ends.*

In the iterative model, representatives of your client can participate in the testing process at the end of each step. The representatives can also suggest changes to the product during development.

If you use the iterative model, you

* don't prepare formal documents, such as the requirement specification document, before you begin development
* don't have to identify all client and system requirements or design the entire product before you start creating code
* have a set schedule and cost for the project and for each iterative step
* repeat each iterative step until you develop a final product that meets all client and system requirements

The iterative model also has its drawbacks. These drawbacks can adversely affect the testing process.  
  
Two significant drawbacks of the iterative model are

**absence of formal documentation**

In the absence of formal documentation, you can't verify requirements accurately. So you write a functional test for each iterative step and ask the developers to create code that can pass the functional test.

**increased testing time and cost**

Compared to the other models, the iterative model can increase testing time and cost. For example, during an iterative step, developers can inadvertently modify a previously approved feature. To prevent such modifications and deletions, you have to spend more effort on testing. This can be a drawback for small-scale projects.

Question

Match each software development model to its diagram.

**Options:**

1. V-model
2. Iterative model
3. Waterfall model

**Targets:**

1. Description of the Model 1 flow chart:  
   The flow chart depicts four rectangular boxes labeled Identify requirements, Create a design, Create code, and Test code. The Test code box is connected to a decision box labeled "Have all requirements been met?". The answer "Yes" takes you to a box labeled End development. And the answer "No" takes you back to the box labeled Identify requirements.   
   Description ends.
2. Description of the Model 2 flow chart:  
   The flowchart depicts six rectangular boxes labeled Identify client requirements, Identify system requirements, Create an overall design, Create a detailed design, Develop product, and Test product. Arrows connect the boxes, starting from the Identify client requirements box down to the Test product box.  
   Description ends.
3. Description of the Model 3 flow chart:  
   The flowchart depicts five rectangular boxes labeled Identify client requirements, Identify system requirements, Create an overall design, Create a detailed design, and Develop product. Arrows connect the boxes, starting from the Identify client requirements box down to the Develop product box. Attached to the first four boxes are four other boxes labeled Plan for acceptance testing, Plan for system testing, Plan for integration testing, and Plan for component testing, respectively. The Develop product box is connected to a box labeled Component testing. This box is, in turn, connected to the box labeled Integration testing. The Integration testing box is connected to a box labeled System testing, which is, in turn, connected to the box labeled Acceptance testing.   
   Description ends.

Answer

*In the iterative model, you perform a series of iterative steps until all your project requirements are met.*

*The waterfall model is a sequential software development model. In this model, you build and then test a software product.*

*The V-model is a customized version of the waterfall model. In this model, you combine the testing process with the software-development process. This ensures that you test each component, feature, and function of your product as it is built.*

**Correct answer(s):**

Target 1 = Option B

Target 2 = Option C

Target 3 = Option A

The V-model and the iterative model are not stringent and you can modify them to suit your requirements. For example, you can use the four test levels – component testing, integration testing, system testing, and acceptance testing – during each iterative step of the iterative model.

You can also customize the order in which you perform the four test levels, regardless of the development model you use. For example, if you're testing a web portal that contains a third-party application, you perform acceptance testing on the third-party application first. Then you carry out system testing for the web portal.

Question

Match each test level to the correct scenario.

**Options:**

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

**Targets:**

1. You are testing the interaction between the component of your software product and an operating system
2. You are testing whether a software product meets users' needs
3. You are testing an assembly that you want to add to your web page
4. You are testing whether a tax-calculator application performs calculations correctly

Answer

*Integration testing helps you determine whether the components of your software product interact correctly with computer hardware and other software. It also helps you test the interrelationships between the components.*

*During acceptance testing, you check whether or not your software product meets all actionable client requirements.*

*During component testing, you check each component of your software product for defects.*

*At the system testing level, you check a complete product and ensure that it meets all client and system requirements.*

**Correct answer(s):**

Target 1 = Option B

Target 2 = Option D

Target 3 = Option A

Target 4 = Option C

**2. Summary**

The development model of any software-development project depends on its requirements. Sequential and iterative models are two types of development models. Testing is an integral part of both these models.  
  
The waterfall model is the simplest sequential development model. In this model, you test a complete product. So you should use the waterfall model only if you've identified the client and system requirements accurately and don't expect to find extensive defects in the product.  
  
The V-model is an improved version of the waterfall model. In this model, you divide the testing activity into four test levels: component testing, integration testing, system testing, and acceptance testing. You use these test levels as you progressively build the product. This makes the testing and fixing processes easy.  
  
When using the iterative development model, you develop your software product in parts, using a series of iterative steps. Testing is a part of each step, and you can involve client representatives in the testing process. This makes it easier to identify bugs.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#top)

Component and Integration Testing

Learning Objectives

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

* *recognize how component testing works*
* *recognize how integration testing works*

**1. Component testing**

In advanced software development models such as the V-model, testing is an elaborate activity and is performed at four levels. Dividing the testing activity into these levels helps you efficiently manage and control the activity. These test levels also make it easier for you to identify bugs and fix them.

Component and integration testing help you locate code-related defects and determine whether the various code segments interact with each other as intended.  
  
You perform component and integration testing during the early stages of software development. So you can correct identified defects with minimal rework and reduce, or even prevent, defects in the integrated software.

Component testing helps you identify errors in each component, such as an object, program, or module, of a software application. After the code for a component is written, it can be tested by the original developer, or passed for testing to another developer. The involvement of another developer brings objectivity and independence to the component-testing process.

During component testing, you don't maintain a formal record of defects. So to avoid oversights later, you or the designated tester should fix all errors as soon as you detect them.

A common approach to component testing is the test-first approach, which is an iterative process. According to this approach, you create test cases before you start developing and testing code. To create the test cases, you use the specifications documents and the blueprint for the software application.  
  
The test-first approach involves repeating three steps until the code passes the tests:

* create test cases
* develop code
* run tests

While you're testing the code of a component, you may also need to check whether the code correctly responds to or calls other components. However, sometimes, you may not have immediate access to these other components.   
  
The unavailability of the other components in a component-testing scenario is similar to the testing of parts at a car-manufacturing plant. For example, when you begin testing the motor of a particular car model, the braking system or the ignition mechanism might not be ready. Here the braking system is a *called* component and the ignition mechanism is a component that calls the motor.

To test the functions of the motor in the absence of a braking system and ignition mechanism, you might first create a temporary, no-frills braking system and ignition. Then you could connect both to the motor. Similarly, to check whether a software component can respond to or call other components, you create two temporary components:

**stubs**

Stubs simulate called components. While testing a software component in isolation, you use a stub in place of each called component. For example, if an component calls an object that is not yet ready, you can use a stub in place of the object.

**drivers**

During component testing, you may also have to create simulated components or test tools known as drivers. A driver acts as a substitute for a component that calls the component you're testing. For example you can use a driver in place of the button that calls the authentication system you're testing.

At the component-testing level, apart from checking the functions of a software component, you also verify whether it meets nonfunctional requirements. For example, you check whether a component can manage memory and storage space efficiently.

Note

*A nonfunctional requirement is one that isn’t related to functionality, but to an attribute such as usability, efficiency, reliability, maintainability or portability.*

During component testing, you also test the robustness of a component. For example, you check how well the component responds to invalid inputs. You also check whether the components require additional testing and perform the tests, if required.

Question

The training department at Easy Nomad Travel, a worldwide travel agency, has developed various training courses over the years. To manage these courses, the agency hires your company to develop a modular course management system. As a testing manager, you're responsible for testing the system. Identify the steps which are necessary to ensure successful component testing.

**Options:**

1. Assign a developer to test each module created by another developer
2. Maintain a formal record of defects
3. Ensure all called components are available
4. Create test cases before you develop and test code

Answer

***Option 1:****Correct. To bring objectivity and independence to the component-testing process, you can ask a developer to test a code segment or module created by another developer.*

***Option 2:****Incorrect. It is not necessary to maintain a formal record of defects during component testing. You fix all defects as soon as you detect them.*

***Option 3:****Incorrect. During component testing, you may not have access to all called components. So to check whether the code you're testing correctly calls other components, you substitute the called components with stubs.*

***Option 4:****Correct. A commonly used approach to component testing is the test-first approach. In this approach, you create test cases before you start developing and testing code.*

**Correct answer(s):**

1. Assign a developer to test each module created by another developer  
4. Create test cases before you develop and test code

**2. Integration testing**

After successful component testing, you move on to integration testing. At this test level, you determine whether software components interact with each other as intended.   
  
During integration testing, you also verify whether the software components interact correctly and efficiently with computer hardware and other software, such as an operating system. This level of testing is also known as component integration testing and is typically conducted by either developers or integrators.

You can also conduct an integration test after the system-testing process. At this stage, you refer to integration testing as system integration testing. This type of testing helps you accurately analyze the interactions between a complete software product and other software systems.

During component and system integration testing, you can test the nonfunctional characteristics of the integrated components and systems. For example, after integration, two components may use a single system resource at the same time. This can negatively affect the performance of the components. You can check for such performance degradation during component integration testing.

Before you plan for and perform integration testing, you should choose an integration strategy. This strategy helps you decide the order in which you integrate the components of a software application. The order depends on the number and types of bugs you expect the application to have.   
  
Based on the integration strategy you choose, you can adopt one of two methods to conduct integration testing:

**big-bang integration testing**

You perform big-bang (or non-incremental, as it is formally known) integration testing on a fully integrated system. You should use this method of testing only if you plan to integrate the software components first and then test the resulting system. Bang-bang integration testing proves successful if the components have few and uncomplicated bugs.  
*Description of the big-bang integration testing animation:  
The animation depicts the pieces of a jigsaw puzzle being assembled to complete the image of a car.  
Description ends.*

**incremental integration testing**

Incremental integration testing is opposite to the big-bang method. In this type of testing, you gradually build and test the system on a component-by-component basis. For example, you test the interactions between any two components before you integrate and test additional components.  
*Description of the incremental integration testing animation:  
The animation depicts the pieces of a jigsaw puzzle being assembled one-by-one to complete the image of a car.  
Description ends.*

Both integration testing methods have their advantages and disadvantages.  
  
Big-bang integration testing ensures that you don't require stubs and drivers to simulate missing components. Yet this method is time consuming because it involves testing an entire system. While conducting such tests, you may be unable to identify the cause of defects quickly.  
  
Additionally, big-bang integration testing allows you to detect defects only late in the development process. Fixing defects at this late stage can considerably increase the defects and expenses of the project, leading to budget overruns.

The advantage of incremental integration testing is that you start testing on a small scale. For example, you first test the interaction between two or three components. This increases your chance of isolating defects.  
  
However, when you conduct tests using incremental integration testing, you need to use stubs and drivers to substitute for missing components. Creating the stubs and drivers can be time consuming, increasing the cost and effort involved in the test process.

In most cases, incremental integration testing is preferable to big-bang integration testing because the advantages of incremental integration testing outweigh the disadvantages.

You can divide incremental integration testing into two types:

**top-down integration testing**

Top-down integration testing involves testing the external features of a software application first and then gradually integrating and testing the internal components. For example, you start by testing the interactions of the graphical user interface (GUI), which isn't called by any other component. If the components the GUI calls aren't ready, you use stubs to simulate them. When you receive the missing components, you perform integration testing on them.   
  
The advantage of top-down integration testing is that you don't need to create drivers. Also, this type of testing helps you easily identify and fix defects in the design of your software application. However, additional effort is required to create stubs.

**bottom-up integration testing**

In bottom-up integration testing, you first test the component at the lowest level in the software application. This would be the component that doesn’t call any other components. You can use drivers to simulate the components that call this component. Similarly, you progressively build the application and test the higher-level components.  
  
This method helps you test the interactions between software components more effectively than the top-down method.

Question

Having integrated the components of the Easy Nomad Travel course management system, you want to perform big-bang integration testing. Before you do so, what should you know about this testing method?

**Options:**

1. Is performed in an incremental manner
2. Involves the use of stubs and drivers to simulate missing components
3. Allows you to detect defects at a late stage of software development
4. Tests the interactions between components and computer hardware

Answer

***Option 1:****Incorrect. Big-bang integration testing involves testing an entire system. You perform this type of test only after you've integrated all the software components.*

***Option 2:****Incorrect. You perform big-bang integration testing on a fully integrated system. Because you don't have missing components, you don't require stubs and drivers.*

***Option 3:****Correct. You can conduct big-bang integration testing only after you've integrated all the components of your software application. This prevents you from detecting defects until all the components are ready.*

***Option 4:****Correct. Big-bang integration testing verifies the interactions between software components and other systems such as the operating system, computer hardware, and other software applications.*

**Correct answer(s):**

3. Allows you to detect defects at a late stage of software development  
4. Tests the interactions between components and computer hardware

**3. Summary**

To accurately identify bugs in your software applications, you can test them at four levels. The first two test levels are component testing and integration testing.  
  
During component testing, you test the code of each component of a software application and immediately fix the bugs you discover.  
  
Integration testing helps you verify the interactions between the components of a software application. You can carry out integration testing in two ways – big bang and incremental. Using big-bang integration testing, you test a fully integrated system. Conversely, incremental integration testing involves progressively integrating components and testing their interactions. You can perform integration testing using either a top-down or a bottom-up approach.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#top)

System and Acceptance Testing

Learning Objectives

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

* *recognize how system testing works*
* *recognize how acceptance testing works*

**1. System testing**

After you have successfully tested the features and functions of each component of a software application, you can test the application as a single entity. This level of testing is called system testing.

During system testing, you ensure that the application meets the requirements you'd identified during the initial phases of the software-development process.

You also test whether the application can withstand potential risks, such as hacker and virus attacks. Additionally, during this level of testing you determine whether the application can optimally utilize system resources. Checking whether the application can interact with the operating systems it supports is also a part of the system-testing process.

System testing is typically conducted by a dedicated testing team. This team will assess the application on the basis of its external features and not the underlying code. To ensure that the implementation details are not known to the testers, they should not have worked on the development of the code.

The objective of system testing is to verify that the system meets specified requirements. You test the application in the simulated environment to do this, checking for any defects as you progress.   
  
You can categorize requirements into

**functional requirements**

Functional requirements refer to the features and functions of the application. For example, an order-processing system needs to display order details to users. So the ability of the system to retrieve the order details from a database is a functional requirement.   
  
You can also consider the security mechanism of an application and the ability of the application to interact with other systems as functional requirements.

**nonfunctional requirements**

Nonfunctional requirements are those that are not related to specific functionality, but to attributes such as usability, efficiency, reliability, maintainability or portability. For example, you consider an application efficient if it retrieves information within a few seconds. So the ability of an order-processing system to retrieve customer details within a few seconds may be a nonfunctional requirement. Although somewhat generic, nonfunctional requirements do need to be identified as part of requirements gathering.

While testing functional requirements, you start with a specifications-based black box approach. Black box testing is so called because it takes no interest in the internal structure of the system or component.

White box, or glass box, is used to describe testing that is concerned with the internal workings of a system. As we’ll see, structural testing is one such testing type.

Note

*To influence the experience of developers, testers, and users and to determine what should be tested, black box and white box testing may be used in conjunction with experienced-based techniques.*

While performing nonfunctional tests, you verify quality attribute requirements. For example, you check whether an application performs as efficiently and reliably as intended. You might determine if the system is as easy to learn and use as you require, and test for issues relating to future maintenance.  You may be testing for how easy it is to install or adapt the system into other environments.

The attributes, and their sub-attributes, tested for during nonfunctional requirements testing, are

* Reliability, comprising robustness, fault-tolerance, recoverability and compliance
* Efficiency, comprising speed, resource utilization, and compatibility
* Usability, comprising comprehensibility, learnability, operability, appeal, and compliance
* Maintainability, comprising analyzability, stability, changeability, testability and compliance
* Portability, comprising adaptability, compatibility, installability, replaceability, and compliance

Question

During system testing, you verify whether an application meets two categories of requirements: functional and nonfunctional. Match each type of requirement to the appropriate category.

**Options:**

1. A graphics-sharing web site allows users to edit their graphics online
2. An accounting application must be capable of being expanded in the future
3. A bank's website runs in all types of web browsers
4. The web site of an airline company must be able to process payments made via a range of credit cards
5. An order-processing system provides buttons and menus that are easily accessible
6. A web form displays transaction details only to authorized users

**Targets:**

1. Functional requirement
2. Nonfunctional requirement

Answer

*Functional requirements are application specific. These requirements refer to the features and functions your client wants in an application. An example of a functional requirement is the ability of a photo-sharing web site to allow users to edit photos. Similarly, the ability of a web site to correctly process credit-card payments and the ability of a web form to display transaction details only to authorized users are functional requirements. Not all web sites and web forms need to satisfy these requirements.*

*Nonfunctional requirements are those that are not related to specific functionality, but to attributes such as usability, efficiency, reliability, maintainability or portability. For example, a system’s readiness to have new features and functionality added at some future date is a maintainability attribute.*

**Correct answer(s):**

Target 1 = Option A, Option D, Option F

Target 2 = Option B, Option C, Option E

Question

Which of the following tests form part of system testing?

**Options:**

1. Testing an application that can support a minimum number of concurrent users
2. Testing an application that works properly on a specific Operating System
3. Testing a component that interacts with other components correctly
4. Testing a module of an application before code is available

Answer

***Option 1:****Correct. System testing tests the whole system as it might be used in a production environment.*

***Option 2:****Correct. System testing tests that the system interacts as intended with its environment.*

***Option 3:****Incorrect. This kind of testing would form part of integration testing.*

***Option 4:****Incorrect. This kind of testing would form part of component testing.*

**Correct answer(s):**

1. Testing an application that can support a minimum number of concurrent users  
2. Testing an application that works properly on a specific Operating System

**2. Acceptance testing**

After successful system testing, you hand over the application to the client. Representatives of the client, such as prospective users of the application or designated testers, then conduct acceptance testing. Instead of checking the application for defects, the testers verify whether the application meets all their requirements, both functional and nonfunctional.

Some clients may also send their representatives to your company to conduct acceptance testing on a product. This type of acceptance testing is known as *Factory acceptance testing*. After a successful factory acceptance test, the representatives take the product to their site and perform another acceptance test there. At this stage, you call the test *Site acceptance testing*.

Note

*Overlapping of system and acceptance testing is common during factory acceptance testing. If an application contains multiple independent subsystems, you don't need to wait for system testing on all the subsystems to be complete. You can perform acceptance testing on any subsystem that has passed a system test.*

For successful acceptance testing, you simulate the environment in which the application will run.  
  
During acceptance testing, you determine whether an application

* meets all the specified requirements
* is ready to be deployed
* adversely affects existing applications and business operations

If an application fails an acceptance test, then unless the customer chooses to accept the defect, it needs to be fixed by developers. For example, if you discover that a new order-processing system causes problems when coexisting with an existing line of business system, then a fix will need to be sought.

You can categorize acceptance testing into various types:

**user acceptance testing**

The prospective users of an application conduct user acceptance testing. During this testing process, they check whether the application allows them to perform business tasks. For example, they check whether an accounting application allows them to generate and print reports.

**operational acceptance testing**

Operational acceptance testing is conducted by system administrators. These administrators try to assess whether the application can withstand malicious attacks, can recover from failures, and can be maintained easily and efficiently. For example, a system administrator may check if a customer-survey form can use back up data to recover from failures.

**contract acceptance testing**

If you sign a contract with the client about the type of application you'll create, the client conducts contract acceptance testing. During this testing process, the client ensures that the application meets the acceptance criteria specified in the contract. For example, if a contract mentions that a web site must allow users to customize each web page, the client verifies this.

**regulation acceptance testing**

To ensure that an application complies with standard government, legal, and safety regulations, the client conducts regulation acceptance testing.  
  
Suppose a company is developing software for US government use. There may be a requirement that the software complies with Section 508 Amendment to the Rehabilitation Act of 1973. Compliance means that the software is accessible to people with disabilities, and that access and use are comparable to access and use by a people with no disabilities. The government department receiving the software would test for section 508 compliance against these criteria.

Two other types of acceptance testing are alpha and beta testing. You perform alpha and beta testing on software applications your organization produces for mass use, for example, commercial off-the-shelf (COTS) applications.

During the alpha testing process, your team and a group of prospective customers check an application to ensure that it meets all specified requirements. You perform alpha testing at your site.

After successful alpha testing, you send the application to another set of prospective customers for beta testing to help simulate a real-world environment. These customers check whether the application meets their needs and report problems to you. Based on their reports, you fix the application and then release it to the market.

Note

*Alpha testing that takes place at the developer’s site is sometimes referred to as factory acceptance testing, while beta testing in the customer’s site corresponds to site acceptance testing.*

Question

The development team at your company has created a transaction-processing system for the call center at Easy Nomad Travel. After conducting a successful system test, you send the system to the client for acceptance testing. The client decides to conduct operational acceptance testing on the system. Identify the characteristics of this type of testing.

**Options:**

1. Is conducted by system administrators
2. Is conducted by prospective users
3. Assesses the ability of a system to resist hackers
4. Verifies whether a system meets safety regulations
5. Checks whether a system meets client acceptance criteria

Answer

***Option 1:****Correct. During operational acceptance testing, system administrators try to assess the robustness of a system. They also check whether they can maintain the system easily and efficiently.*

***Option 2:****Incorrect. The prospective users of a system conduct user acceptance testing. During the testing process, they check whether the system can help them perform required tasks.*

***Option 3:****Correct. During operational acceptance testing, the administrators try to assess whether a system can withstand malicious attacks and recover from failures. They also check whether their security guidelines and user-management criteria can be applied to the users of the system.*

***Option 4:****Incorrect. Regulation acceptance testing verifies whether a system complies with standard safety, government, or legal regulations.*

***Option 5:****Incorrect. To check whether a system meets acceptance criteria, representatives of the client conduct contract acceptance testing. The acceptance criteria are part of a contract, which the client and representatives of your organization jointly sign.*

**Correct answer(s):**

1. Is conducted by system administrators  
3. Assesses the ability of a system to resist hackers

Acceptance testing is often the last test level. However, you can also perform acceptance testing at other stages of the software-development process:

**during component testing**

During component testing, you can conduct acceptance testing on a component to verify it performs as expected.

**during installation or integration**

You subject a COTS application to acceptance testing while you’re installing it or while you’re integrating it with another application.

**before system testing**

If you add a new feature to an existing application, you first perform acceptance testing on the feature. Then you conduct system testing on the entire application.

Question

You've handed over the course management system to representatives of Easy Nomad Travel. The representatives want to run an acceptance test on the system. What should they do?

**Options:**

1. Ask a group of prospective users to perform the test
2. Ask the developers at your company to perform the test
3. Check each component of the system for defects
4. Check whether the system is ready to be deployed

Answer

***Option 1:****Correct. A group of prospective users conducts acceptance testing on a system and checks whether the system meets their requirements.*

***Option 2:****Incorrect. The developers who created the application should not be involved in acceptance testing although the customer might commission acceptance testing on their behalf by professional testers from the same company.*

***Option 3:****Incorrect. The client representatives don't perform acceptance testing primarily to look for defects in a system. They perform acceptance testing to verify whether the system can be deployed and meets their key requirements, both functional and nonfunctional.*

***Option 4:****Correct. During acceptance testing, representatives of the client check whether a system is in a satisfactory state and is ready to be deployed. In doing this, they check whether the system can adversely affect existing applications and business operations.*

**Correct answer(s):**

1. Ask a group of prospective users to perform the test   
4. Check whether the system is ready to be deployed

**3. Summary**

Before you deliver a fully integrated application to your client, you perform system testing on the application. This is the first time you test the application as a whole. System testing helps you determine whether the application meets the requirements you'd agreed upon with your client.   
  
Once the application has completed system testing successfully, acceptance testing is carried out for, or on behalf of, the client. The client representatives may carry out the tests in two stages: first at your site (alpha testing) and then at the client’s (beta testing).

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#top)

Testing to Reveal Defects

Learning Objective

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

* *evaluate software defects and test levels*

**1. Exercise Overview**

In this exercise, you're required to evaluate software defects and test levels. First, you need to evaluate given defects and determine which test levels can help detect them. Second, you need to evaluate given test levels and determine which defects the test levels can reveal.

This involves the following tasks:

* recommending test levels
* evaluating test levels

**2. Recommending test levels**

Your company is developing an online-banking web site for Poseidon Bank. Two significant features of the web site are an innovative user interface and an authentication system.   
  
The user interface should allow users to log on and effortlessly navigate through the web site. The purpose of the authorization system is to verify the user credentials and allow authenticated users to view their account details.   
  
You and your team members have been assigned the task of testing the web site.

Question

During which level of testing would representatives of a client typically note a system failure, for example difficulty in reinstating a web site after it has been hacked?

**Options:**

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

Answer

***Option 1:****Incorrect. During component testing, you check the features and functions of each component of an application separately. The representatives of your client are rarely involved in the component testing process.*

***Option 2:****Incorrect. Integration testing helps to verify the interactions between the components of an application. Even if client representatives participate in integration testing, this type of defect is unlikely to be exposed.*

***Option 3:****Incorrect. Before you deliver an application to your client, you assess the features and functions of the entire application using system testing. Typically, the representatives of the client don't participate in system testing.*

***Option 4:****Correct. Acceptance testing is conducted by the representatives of your client. During this testing process, client representatives verify whether an application meets all their requirements and is easy to maintain.*

**Correct answer(s):**

4. Acceptance testing

Question

You want to check whether the authentication component of the web site erroneously grants access to users who provide invalid input. Which testing level helps you achieve this objective?

**Options:**

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

Answer

***Option 1:****Correct. The authentication system is one of the components of the Poseidon Bank web site. To verify whether this component can distinguish between valid and invalid input, you conduct component testing.*

***Option 2:****Incorrect. During integration testing, you don't check the internal workings of a component. Instead, you check whether the component interacts with other components and systems as intended.*

***Option 3:****Incorrect. You don't test individual components during system testing. In this type of testing, you conduct a system test to check an application as a whole and ensure that the application meets client requirements.*

***Option 4:****Incorrect. Acceptance testing is performed by the representatives of a client. Rather than look for defects in an application, these representatives verify whether the application meets all their functional and nonfunctional requirements.*

**Correct answer(s):**

1. Component testing

Question

Users of the web site should be able to locate links and navigation controls easily. Which test level helps you verify this requirement?

**Options:**

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

Answer

***Option 1:****Incorrect. Ease of use and effortless navigation are external features of an application. Component testing can't help you test these features. It only helps you test the code for individual components of the application.*

***Option 2:****Incorrect. Integration testing helps you verify the interactions between individual components of an application.*

***Option 3:****Correct. Ease of use is one of the nonfunctional requirements of an application. You verify this requirement during system testing.*

***Option 4:****Incorrect. Acceptance testing helps verify whether an application can be deployed by the client. This level of testing is conducted directly by, or on behalf of, the client.*

**Correct answer(s):**

3. System testing

**3. Evaluating test levels**

Your company develops gaming software for a wide range of customers and sells the software through its worldwide retail outlets.  
  
As a testing manger, you supervise the testing of the software. You're currently supervising the testing of a product aimed for children in the age group 7 through 10. The product will allow users to change display preferences and color schemes. Also, the product will provide four levels of complexity during each gaming session.

Question

You ask developers to perform component testing on each component they develop. What type of defect can they discover during component testing?

**Options:**

1. The processor of the product cannot resolve user input
2. A logic defect in the user-level management system
3. The product is incompatible with Microsoft Windows Vista
4. The target audience users find the graphical user interface confusing

Answer

***Option 1:****Incorrect. The processor receives user input from another component, the user interface. You can verify whether it can interact correctly with the user interface during integration testing.*

***Option 2:****Correct. You use component testing to detect defects in a particular component, such as the component that manages user levels.*

***Option 3:****Incorrect. You verify the compatibility of a product with an operating system during system testing. You can make such verifications by simulating the environment in which the product will run.*

***Option 4:****Incorrect. The discovery that the target market users find the graphical user interface confusing will typically arise at the acceptance testing level, in particular, during beta testing.*

**Correct answer(s):**

2. A logic defect in the user-level management system

Question

You are conducting system testing, and have simulated the environment in which the intended users will run the product. Which types of defects are most likely to be exposed during system testing?

**Options:**

1. Too few of the intended users are able to reach level four of the game
2. The output of the user interface is corrupted after it's transferred for processing
3. Sessions become slow for users at the third and fourth levels of a gaming session
4. The product is not compatible with computers that have less than 256 MB of memory

Answer

***Option 1:****Incorrect. The ability of the user population to master the product as expected is tested during acceptance testing, not system testing.*

***Option 2:****Incorrect. The user-interface is one of the components of the product, and its output is corrupted while it is interacting with another component. You test the interactions between components during integration testing.*

***Option 3:****Correct. A nonfunctional requirement of any software product is that it responds to users within a reasonable period. You can verify such requirements only after you integrate the product and conduct performance testing.*

***Option 4:****Correct. To verify the compatibility of a product with a given system, you need to run it on that system. You simulate the environment in which a product will actually run during system testing. So you can use this type of testing to verify the compatibility of the product.*

**Correct answer(s):**

3. Sessions become slow for users at the third and fourth levels of a gaming session  
4. The product is not compatible with computers that have less than 256 MB of memory

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#top)

Functional and Non-functional Software Testing

Learning Objectives

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

* *recognize how functional software testing works*
* *recognize how non-functional software testing works*

**1. Functional software testing**

Testing happens at all phases of the product development life cycle. At every stage of development, testing reveals defects that need to be fixed to assure quality. When testing a software product at various levels, merely testing the functionality of each component or system is not always sufficient.

You define test objectives for the various levels of development. To meet the overall test objectives, you focus your testing on specific objectives at each level. To meet these specific objectives, you use different test types.  
  
These test types are typically a set of test activities, focused on a specific test objective, used to test different components and systems. Different tests enable you to meet each specific objective in different test phases, whether you’re testing a function or testing modifications made to the software.

There are different test types:

* functional testing – tests the functionality of a selected component
* non-functional testing – tests the behavioral, or the quantified characteristics, of the systems and software
* structural testing – tests the structural aspects of the component or system
* changed-based testing – including regression and confirmation testing that involve the re-run of tests to ensure that the software is working correctly following changes

Functional testing is the process of testing a software product to determine its specified behavior or functionality. As a functionality tester, you focus on the software's capability to provide functions that fulfill stated and implied needs under specified conditions. In other words, you test the function that the component or system needs to perform.

For example, consider software that allows people to book tickets online. To test the functionality of this software, you would have to test whether the software accepts information requests from travelers, checks for ticket availability and reports back, accepts applications and returns tickets, and finally records the transactions in a database.

Based on the International Organization for Standardization (ISO) quality standard 9216 that regulates testing, functional testing is performed for various quality characteristics, such as

**suitability**

Testing suitability involves testing the capability of the software product to provide an appropriate set of functions for specified tasks and user objectives. Suitability testing determines if the product performs as expected for its intended use. For example, a spreadsheet should provide you with an appropriate set of functions to perform financial, logical, mathematical, and trigonometrical calculations.

**interoperability**

Testing interoperability involves evaluating the capability of the system to interact with other specified components or systems.

**accuracy**

Testing accuracy involves ensuring that faulty products do not leave the production line and cause errors during beta testing.  
  
For example, if a word count feature in a word processing application does not function properly, accuracy testing would detect this defect before a beta user discovers it.

Based on the standard, functional testing is also performed for various quality characteristics, such as

**security**

Testing security involves investigating the functions relating to prevention of unauthorized access to software and data, either intentionally or unintentionally. It also involves detection of malicious outside threats, such as viruses.  
  
For example, if readers can leave comments on your corporate blog, you will have to ensure that user input validation is performed in the comments area. There is the possibility of a malevolent user typing in malicious code, which gets executed automatically. Security testing would be able to capture this loophole in the code.

**compliance**

When you test compliance, you determine whether the system adheres to certain specified criteria, such as standards, conventions, regulations, or laws.  
  
For example, suppose that your organization's information security policy requires that employee system CD drives be disabled at all times. Code that is run every morning during system boot to verify that the drive is disabled would be checked during compliance testing.

To perform functional testing, you derive your test conditions and cases from the requirements specification, functional specification, or from use cases. You design a model, such as a state transition model, a process model, or a plain language specification, as part of the test design. In all cases, the functional tests performed across all test levels would look at specific functions in a system, including undocumented or implicit functions.

Because only the program specification is considered – not the design or implementation of the program – this type of testing is also called specifications-based or black box testing.

Note

*Black box testing can be either functional testing or non-functional testing and is performed without reference to the internal structure of the product.*

Two approaches to functional testing are:

**requirements-based testing**

Requirements-based testing designs tests based on a functional requirements specification for the system. To decide the items to test, you can use the table of contents of the requirements specification and prioritize the requirements based on the risk criteria. This ensures that the most important tests are included in the testing effort.  
  
For example, your company is revamping its HTML-based corporate web site. Your job is to test if the ASP.NET-based web applications display properly in a web browser that supports HTML. You refer to the requirements specification to create the test cases.

**business-process-based testing**

Use cases provide a basis for test cases from a business perspective because they often use business processes as a starting point. Business-process-based testing uses knowledge of the business processes to describe the scenarios involved in the day-to-day business use of systems.  
  
For example, your company's travel reimbursement system may have a policy that determines reimbursements for employees traveling on business.

Question

Match each characteristic against its function.

**Options:**

1. Accuracy
2. Suitability
3. Interoperability
4. Security

**Targets:**

1. Tests if the product performs according to specifications
2. Tests if the product is shielded from threats, such as viruses and other malicious software.
3. Tests to ensure that the faulty products are not passed to users
4. Tests if the product interacts correctly with other components or systems

Answer

*Suitability testing verifies that the product performs according to specifications and as expected for its intended use.*

*Security testing investigates the functions relating to detection of threats, such as viruses, from malicious outsiders.*

*Accuracy testing ensures that faulty products do not leave the production line and cause errors during beta testing.*

*Tests if the product correctly interacts with other components and systems as specified.*

**Correct answer(s):**

Target 1 = Option B

Target 2 = Option D

Target 3 = Option A

Target 4 = Option C

Question

You are the tester for a web-based banking application where financial data is requested from the user and stored in a confidential database. How would you perform functional testing for this application?

**Options:**

1. Verify that unauthorized access to data is prevented
2. Test the quantified characteristics of the systems and software
3. Ensure that any shortfalls in the required level of precision of the application’s results and effects are detected
4. Conduct tests that verify the impact of a modified operational system on the existing environment

Answer

***Option 1:****Correct. Because customer financial information is confidential, you perform security testing to ensure its safety.*

***Option 2:****Incorrect. You test the quantified characteristics of the systems and software in non-functional testing.*

***Option 3:****Correct. You perform accuracy testing on the application to ensure that an application that does not meet required levels of precision is not released, particularly because it deals with sensitive financial data.*

***Option 4:****Incorrect. You test modifications and verify the impact of these modifications on the existing environment in maintenance testing.*

**Correct answer(s):**

1. Verify that unauthorized access to data is prevented  
3. Ensure that any shortfalls in the required level of precision of the application’s results and effects are detected

**2. Non-functional software testing**

As well as testing the functional aspects of a software system, you must also test the non-functional aspects of the system. Suppose you are testing the operation of a mathematics package. You finish testing the functional aspects, such as fomulae and functions, and then move on to how well the package performs. Here, you would test how the system manages performance, load, stress, portability, and so on. This type of testing, where you measure how well or fast a system works, is called non-functional testing.

In non-functional testing, which can be performed at all test levels, the behavioral characteristics of systems and software are tested and quantified on a varying scale.

For example, you are testing a gaming application that will be hosted online and played by gamers simultaneously across the world. During functional testing, you determine what the application does, test its suitability for the web, verify compliance with online gaming standards, and so on. In contrast, when doing non-functional testing, you will test how the application performs, the load that it can handle, the speed with which it performs, and other aspects.

Non-functional testing comprises various types of testing:

* performance testing – tests the degree to which a system fulfills its specified functions within given processing time and throughput rate constraints
* load testing – measures the behavior of a system with increasing load
* stress testing – evaluates a system at and beyond the boundaries of its specified requirements
* usability testing – tests how easily a user can perform a specific task
* maintainability testing – tests how easily a product can be modified in the future
* reliability testing – tests how reliably a product performs over a given period of time
* portability testing – tests how easily a system can be transferred from one platform to another

According to ISO 9216 there are five characteristics covered by non-functional testing. These are further divided into sub-characteristics.  
  
The first two characteristics are

**reliability**

A software product is reliable when it performs its required functions under stated conditions. Reliability testing tests if the product meets the standards of reliability. Reliability testing can be defined further into maturity, fault-tolerance, recoverability, and compliance. For example, a server could be tested for robustness by increasing the number of users that access the server until it crashes.

**usability**

A software product is said to be usable if the user easily understands and likes the interface, likes the product itself, and finds it easy to operate. Usability testing tests the extent to which the software product meets these requirements under the specified conditions. Usability testing is divided into understandability, learnability, operability, attractiveness and compliance. For example, a graphical user interface could be tested to determine ease of use and navigation.

The other three characteristics are

**efficiency**

Efficiency is the capability of the software product to provide appropriate performance, relative to the amount of resources used under stated conditions. Efficiency testing checks if the software is efficient. Efficiency testing is divided into performance, resource utilization, and compliance. For example, a web page could be tested to check if it loads quickly or not.

**maintainability**

The ease with which a product can be modified to correct defects, meet new requirements, make future maintenance easier, or be adapted to a changed environment is called maintainability.  Maintainability testing is divided into analyzability, changeability, stability, testability, and compliance.

**portability**

When a software product can be transferred easily from one hardware or software environment to another, it is said to be portable. Portability testing is the process of testing to determine the portability of a software product. Portability testing is divided into adaptability, installability, co-existence, replaceability, and compliance. For example, checking the software code to verify if the application has been created to be platform independent or not.

Question

Match the testing types with their explanations.

**Options:**

1. Reliability testing
2. Usability testing
3. Efficiency testing
4. Maintainability testing
5. Portability testing

**Targets:**

1. Test the ease with which a product can be modified to meet requirements
2. Test how easily a user understands an application's interface
3. Test how easily a software product can be transferred across platforms
4. Test the product's ability to perform required functions under stated conditions for a specified period of time
5. Test capability of the software to perform as specified

Answer

*Maintainability testing tests the ease with which product modification can be undertaken to correct defects, meet new requirements, or adapt to changes.*

*Usability testing tests how comfortable a user is with an interface and how easy he finds it to operate when used under specified conditions.*

*Portability testing is the process of testing to determine the portability of a software product.*

*Reliability testing is the process of testing if the product meets the standards of reliability.*

*Efficiency testing checks the capability of the software product to provide appropriate performance, relative to the amount of resources used under stated conditions.*

**Correct answer(s):**

Target 1 = Option D

Target 2 = Option B

Target 3 = Option E

Target 4 = Option A

Target 5 = Option C

**3. Summary**

Test types are activities that help you in testing systems of components to check if they meet specific test objectives. There are a number of test types.  
  
Functional testing checks if the software product under test meets all its functional specifications. It also checks if the software product has the characteristics of suitability, interoperability, security, accuracy, and compliance.  
  
Non-functional testing checks certain quality characteristics of the software product. This includes checking for performance, reliability, portability, usability, and maintainability.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#top)

Structural and Changed-based Software Testing

Learning Objectives

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

* *recognize how structural software testing works*
* *recognize how change-based software testing works*

**1. Structural software testing**

Every time a system or a component is developed, you test it to verify that the system not only works as specified but also as efficiently as required. To aid you in this testing, you are provided with test objectives and test types.   
  
The test objectives will define the test types which are used, such as functional testing or black-box testing, non-functional testing, structural testing, and regression testing. Each test type has specific types of test objectives pertinent to that type of test.

You carry out functional testing to check that the application does what the functional specification requires. Since functional testing is not directly concerned with the inner workings of the software, and only on what it actually does, it is sometimes referred to as black-box testing.

Structural testing on the other hand is concerned with the internal architecture and workings of the software, and for that reason is often referred to as white-box, or glass-box, testing.

Structural tests can be carried out at all test levels. However, they are mostly performed at the lower test levels such as at the component and the component integration level.   
  
At the component level, the tests are based on the structural aspects of the system such as code in a program. And at the component integration level, the tests may be based on the architecture of the system, such as a calling hierarchy.

Structural tests measure the amount of testing done by checking the coverage of a set of structural elements or coverage items.

In structural testing, in addition to testing the coverage of a set of structural elements or coverage items, the internal system design is considered and the tests are based on the logic of the application's code. But in functional testing, the tests are based on the requirements and the functionality.

For instance, when testing a data object, functional testing tests the working of the data object. And structural testing verifies if the data object has the specified data fields with correct data types.  
  
So, in structural testing, you are required to have knowledge of the internal working of software and the code. Regardless of the specifications, you are required to test each command in the code.

To ensure that you are testing your program adequately, you need to execute all the elements such as statements, branches, conditions, and decisions in the code one by one.

The coverage achieved in structural testing is expressed as a percentage of the items being covered. There are a variety of tools and methods available to support code coverage measurement at component and integration testing levels.  These tools measure the percentage of executable elements.

If the coverage is not 100%, then more tests may be designed to test those items that were missed, and thereby, increase the coverage.

You use various structure-based techniques to implement structural testing. These techniques are also called white-box techniques. They are based on an analysis of the structure of a component or a system.   
  
Two examples of code-related structural testing techniques for code coverage at the component level include statement testing and decision testing.

White-box techniques can also be used at various levels, such as

**component**

At the component level, the structure is that of the code itself, such as in the statements or decisions.

**integration**

The structure at the integration level could be a call tree where modules call other modules.

**system**

At the system level, the structure may be a menu structure, a business process or a web page structure.

Question

Which of these are accurate statements when applied to structural testing?

**Options:**

1. Is also referred to as white-box testing
2. Is also referred to as black-box testing
3. Is used to measure the coverage of functional requirements that have been satisfied
4. Is used to measure the percentage of code elements that have been executed.

Answer

***Option 1:****Correct.  Because structural testing is concerned with the internal structure of a component or system, it is also known as white-box, or glass-box, testing.*

***Option 2:****Incorrect. Black box testing is so called because it is not concerned with the inner workings of a component or system, unlike structural testing.*

***Option 3:****Incorrect.  Functional testing, not structural testing, is primarily used to test whether functional requirements have been met.*

***Option 4:****Correct. Structural testing incorporates code coverage measurement to determine the percentage of executable elements that are activated during testing.*

**Correct answer(s):**

1. Is also referred to as white-box testing  
4. Is used to measure the percentage of code elements that have been executed.

Question

You are testing an application. Which of these steps will you do while performing structural testing on the application?

**Options:**

1. Measure the amount of code used in a system
2. Test all the elements in the code
3. Use structure-based techniques to test components
4. Complement functional tests by measuring the number of tests performed
5. Check how much of the item being tested is covered by the test

Answer

***Option 1:****Incorrect. While performing structural testing, you measure the amount of a system or component that is covered by the test.*

***Option 2:****Correct. In structural testing, you execute all the elements in the code such as the statements, branches, conditions and decisions in the code.*

***Option 3:****Correct. At component level testing, you use structure-based techniques such as statement testing and decision testing.*

***Option 4:****Incorrect. You would complement functional tests by measuring the extent of the item that has been tested, not the number of tests performed.*

***Option 5:****Correct. To measure the thoroughness of the tests performed earlier, your structural tests check the coverage achieved by those tests.*

**Correct answer(s):**

2. Test all the elements in the code  
3. Use structure-based techniques to test components  
5. Check how much of the item being tested is covered by the test

**2. Changed-based software testing**

Testing is carried out at every stage of the product development to discover the defects, if any, and fix them. This ensures that the product is defect free and works as specified.   
  
Every time you find a defect and fix or debug it, you need to retest the debugged component by re-executing the test cases that failed the last time. The product is considered free of known defects only when it passes the retest.  
  
This type of retesting is called confirmation testing.

Confirmation testing by itself does not guarantee a quality product. Even after confirmation testing, because of fixing a bug, or any other modification such as the addition of a new feature, a debugged or new version of the software can lose its original functionality or a part of it. When this happens, the new version is said to have regressed with respect to former versions, and is defective again.

A systems or component that has had changes carried out needs to be tested again to ensure that none of its functionality was modified unintentionally. A set of tests are designed to demonstrate that despite the changes that were made, the system works as expected. These tests are called regression tests and the testing is called regression testing.

Like confirmation testing, regression testing involves executing test cases that were executed before. However, confirmation testing is performed on the items that failed the tests the last time they were executed, to confirm that the developers have fixed the defects. And regression testing is performed to ensure that previously working parts of the system have not had faults introduced as a result of these fixes.

There are three types of regression:

* local regression – a change or a bug fix in the existing software creates a new bug
* exposed regression – a change or a bug fix in the existing software reveals an existing bug
* remote regression – any change or a bug fix in one area triggers an error in another area of the system

While performing regression testing, most organizations use a regression test suite, which is a set of test cases used for regression testing at each level. And where the postcondition of one test is used as a precondition for the next one. Also because the same test cases are used every time, the test cases can be used for test automation. The test cases perform an overall testing of each of the most important functions in a system. They are executed every time software is updated or the environment is changed.

It is important to maintain the test suite so it stays updated with the latest software version. So every time new functionality is added to software, new regression tests should also be added and the older and irrelevant tests removed.  
  
In a situation where the regression test suite becomes very large, and the tests need to be executed, a subset of the test cases can be chosen. This is because it is difficult to execute all the tests manually or simultaneously. In such cases, or in cases when the test cases have not captured a defect in a long time, repetitive tests can be combined or, if needed, even eliminated, but with care.

Note

*Unlike the other types of testing, regression testing can be carried out only at the very end, after the last code that was changed is tested and quality passed.*

It is also important not to omit regression testing because the risk involved is huge. Consider a case where you have added a new feature to an existing product.  
  
While functional tests help you verify the functionality of the features, non-functional tests test the behavioral characteristics of software, which are quantifiable on a varying scale. And structural tests measure the amount of the developed component that has been exercised by testing.  
  
After running the new code through all these tests, you find it passes the tests and is free of known defects. You release your product to the market confidently. And then suddenly find out that faults have surfaced in parts of the system that were previously working OK.

There are a number of strategies available to capture defects in the regressed software to avoid such situations. The main strategies are

**repeating all the tests**

If you have sufficient time and resources to cover critical risks, repeat all your tests after the last change to the code. You should be able to find all the important regression bugs. However, if you are repeating these tests for large, complex systems, you can use automation.

**repeating some of the tests**

It is not always possible to repeat all the tests. Executing all the test cases for a large and complex software product may require too much time and too many resources to be practical every time the product is updated. Even if you consider automation, you may find that full automation is impossible or impractical. In such cases, you can repeat some tests only.

Other strategies are

**using cross-functional tests**

You use cross-functional tests to capture accidental regressions. Otherwise, because you are not going to be running all the tests again, you are likely to miss regressions that may occur in unanticipated areas.

**releasing the product in various phases**

You can also choose to release your updates on the product in phases. You will need to phase your release so that you have more time to ensure that your product is thoroughly tested.

**having other users testing the product (Beta testing)**

When you send your product for a Beta test, it will be your users who will be testing your product for you before the commercial release.  
  
If you find yourself bogged down with schedule pressure, instead of releasing poorly tested emergency patches, it will be a good idea to release smaller but well tested patches to only those users who need them. You can then roll the final and completed patches into maintenance releases instead. Doing this will ensure that the regression risk increase for your users is only short term.

If you are using the strategy of repeating tests, you will need to decide which of the tests to repeat. To decide which tests to repeat, you can use techniques, such as

**traceability**

You use the traceability technique when you want to check whether any requirement, design element, or quality risk is affected by the software modification. You would have a set of tests related to the behavioral descriptions of the system. You trace back to these tests and re-execute them.

**change analysis**

When you want to analyze how changes in one part could trigger off changes in other portions of the system, you use the change analysis technique. So you look at the structural descriptions to find the answers. To be able to do this, you need to have an indepth understanding of the code and the system's design.

**quality risk analysis**

If you want to decide what to retest based on business risks, you should use the quality analysis technique. Even though defects are unlikely here, you will have knowledge of the areas that are prone to high business risk. Using traceability and change analysis techniques will only help you select tests based on technical risks.

Question

Match the four test types with their definitions.

**Options:**

1. Functional testing
2. Non-functional testing
3. Structural testing
4. Regression testing

**Targets:**

1. Is based on an analysis of the internal structure of a component or system
2. Tests the working of a component or system and how it interoperates with other specific systems
3. Confirms that the system works as expected even after changes
4. Tests the behavioral aspects of the system

Answer

*Structural tests measure the amount of a component or system that has been exercised by testing against the total amount possible, which is known by reference to the internal structure.*

*When you perform functional testing, you focus on the testing of a function that the component or system needs to perform.*

*When you perform regression testing, you create a set of tests that demonstrate that the system works as expected despite the modifications made.*

*When you perform non-functional testing, you test the characteristics of systems and software that can be quantified on a varying scale.*

**Correct answer(s):**

Target 1 = Option C

Target 2 = Option A

Target 3 = Option D

Target 4 = Option B

Question

You are testing courses for an online education provider who has converted all Director-based courses to Flash. You do not have the time to repeat all the tests. How would you perform your regression tests?

**Options:**

1. Repeat all the tests nevertheless
2. Repeat some tests only
3. Automate all the tests
4. Use techniques such as traceability, change analysis, or quality risk analysis

Answer

***Option 1:****Incorrect. Repeating all the tests is not only impractical, it is impossible considering that you do not have the time to repeat all of them.*

***Option 2:****Correct. Even if the best thing would be to run all the tests again, when you do not have the time for doing it, you repeat only some of the tests.*

***Option 3:****Incorrect. Automating the tests is not possible without a test suite.*

***Option 4:****Correct. When you've decided to repeat some tests only, you can use different techniques to decide which tests to repeat.*

**Correct answer(s):**

2. Repeat some tests only  
4. Use techniques such as traceability, change analysis, or quality risk analysis

**3. Summary**

Structural tests are based on an analysis of the internal structure of a component or system, and measure the amount of structural elements or coverage items that have been exercised by testing.  
  
The structural aspects could be the code in an application or the logic behind a decision. While these tests can be carried out at all levels, they are mostly performed at the component and integration levels.  
  
Change-based testing is concerned with testing that is carried out on foot of changes made to a component or system. Confirmation testing is performed on items that failed the tests the last time they were executed, to confirm that the developers have fixed the defects. And regression testing is performed to ensure that previously working parts of the system have not had faults introduced as a result of these fixes.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#top)

Maintenance Software Testing

Learning Objective

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

* *recognize how maintenance software testing works*

**1. Maintenance software testing**

To ensure a product is free of known, unacceptable defects, and meets user requirements, you need to perform testing at every stage in the product development life cycle. You have to carry out functional and non-functional testing in the initial stages of the development and then follow it up with structural, confirmation, and regression testing in the later stages.

After the product is deployed and in use for a number of years, it may require maintenance work to be carried out on its components. This work may be to correct defects, to improve performance, or to adapt the product to a modified environment. To confirm that the modified product is once again free of known defects, you need to retest the modified components. This is termed maintenance.

And this type of retesting when you test a system, which is in operation in a live environment, is called maintenance testing. Maintenance testing not only tests the changes to the operational system, but also the impact of this modified system on the existing environment.

The main reasons for using maintenance testing include

**modifications**

Modifications refer to any enhancements and emergency corrections made to an operational system. They also include environment changes such as operating system or database upgrades and patches for rectifying any vulnerabilities. Maintenance testing for modifications involves verifying that these changes do not adversely affect the system.  
  
For example, your organization has upgraded its database and installed a patch to a newly discovered operating system vulnerability. You have to ensure that the data is not lost and that these changes do not affect any other functionality in the system.

**migrations**

Migrations refer to a transfer of a system from one platform to another. Maintenance testing for migrations involve operational testing of the new environment and the modified software. This operational testing is conducted to evaluate the working of the migrated system or component in its new operational environment, which typically consists of software installed at the users site.  
  
For example, your organization is migrating to a new environment. After migration, the modified software may not work as required. You have to ensure that the new environment and all software specially modified for it, work fine.

**retirement of the system**

Maintenance testing for the retirement of a system refers to testing of data migration. Or, if data is to be retained for long periods, archiving is considered.  
  
For example, your organization is retiring a server and replacing it with another one. All the data stored in the old server has been transferred to the new server. Some of the tables were merged in the process. You have to check that all the data was transferred accurately and into the specified rows and columns of the new tables.

Of the main reasons for maintenance testing, modifications are the ones most frequently occurring in organizations, and therefore very important. From the view point of testing, modifications can be classified into

**Planned modifications**

Planned modifications account for around 90% of the maintenance work. They are structured modifications and are decided well in advance.  
  
Planned modifications include three types of modifications. Perfective modifications refer to changes such as including additional features or enhancing performance of the system. Adaptive modifications refer to changes such as the product being retired or migrated to a new platform or new environment, new software, or new legislation. Corrective planned modifications refer to changes such as deferred defects being finally corrected.

**Unplanned modifications**

Unplanned modifications occur when defects arise suddenly and the consequent malfunctions require an immediate solution. For example, a server being patched when a security vulnerability is discovered.  
  
By carrying out a risk analysis on the operation system, you can be prepared to minimize the impact of most unplanned modifications by, for example, having those tests that will most likely be required already designed and ready.

During maintenance testing, two fundamental tests are carried out to ensure the quality of the product:

* confirmation tests – to verify that changed software is functioning correctly after the changes have been carried out
* regression tests – to check that unchanged software is continuing to work as expected following changes being carried out on different parts of the system

Ideally, maintenance testing should test all the changes made to the system and the working of the entire system itself. However, this may not always be cost effective or even possible. So to reduce the amount of regression testing, you should always first determine the parts of the system that could be affected because of the maintenance work. Analyzing this impact of the changes on the system is called impact analysis.

Maintenance testing often makes use of original test specifications with some updates applied. An impact analysis helps in assessing the amount of updating required to the original test specifications.  
  
However, it may be difficult to conduct an impact analysis for a system which has already been released. This is especially the case when a product's specifications are out of date, or unavailable, and/or the original development team has moved on to other projects, or is no longer with the company.

Impact analysis enables you to focus your regression test activities where they are most needed.  The process involves:

**performing a risk analysis**

The function of a risk analysis is to establish which of the areas of a software product constitute the greatest risk to the operational services. When you do not have the time to repeat all the tests, you perform a risk analysis to help you determine which areas to focus on during regression testing.  
  
For example, you perform a risk analysis on a software system that malfunctions to help you decide where to focus first and what to do next and where.

**checking for the product's original test specifications**

If you have the test specifications from the original development process, you can reuse them for the regression tests. You do this by adapting them for changes to the system. For instance, you may just change the expected results for the original tests. This helps in reducing the overall regression testing time.  
  
This also helps you decide if you need to build additional tests for specific areas, which is often the case for enhancement or extension projects or for updates for automated test sets that support regression testing.

There are some specific differences between maintenance and regular testing for new products. The differences are due to factors, such as

**test types**

Maintenance testing can include all test types that a new product undergoes, although often only a subset of test types is required.  For example, usability testing is unlikely to be required if changes affecting the user experience have not been made.  In most cases however, regressions and confirmation tests form part of maintenance testing. Maintenance testing is also different from maintainability testing, which is a test to determine how easily a system can be maintained. It also tests how quickly a system can be modified to meet new requirements, adapt to a changed environment, or correct existing defects, and make future maintenance easier.

**test triggers**

While testing for new products is carried out as part of a project, maintenance testing is carried out as an activity in a regular organization, only when there are modifications to the system, or migration, or retirement activities.   
  
This makes maintenance testing more prone to challenges such as resource availability, flexibility, and competition from other activities when compared with regular new product testing.

**issues faced**

When systems are maintained for a while, a number of defects come to light. Specifications are needed, but unlike in the testing for a new product, they are often unavailable.  
  
Also whenever there is an extension or addition, specifications have to be created. For this you can use documentation similar to the original product specifications. But this poses a problem too as you may end up having many test cases executing the same scenario. And because no traceability exists, if an incident is found, it is difficult to trace it back to the actual defect.

From a test management perspective, a somewhat different path is followed for maintenance testing as against new product testing.  
  
Suppose you are the test manager carrying out the fundamental test activities for a new product. Upon the receipt of an application for a product test, you produce a test plan using the application as a basis. Next, you receive test specifications and create test cases, which you execute upon receiving the test object. Finally, upon completion of the testing, you preserve the testware.

Suppose you are carrying out the fundamental test activities for a maintenance product instead. You will follow the same process followed for a new product, except for when it comes to the test specifications. This is because unlike in regular product testing, test specifications may not be available. Especially in the case of enhancements or extensions.  
  
So, if test specifications are not available, and if it is impossible to compile them again, you use an alternative test basis such as a test oracle. The test oracle contains information that is used to determine expected results to compare with the actual results of the tested product. The oracle may be anything from the current system in use to a tester's knowledge, as long as it is not code.  
  
Finally, upon receipt of the test object, you execute the new and modified tests and perform the regression tests, and then preserve the testware.

Question

Match the definitions of the terms used in maintenance testing to the corresponding term.

**Options:**

1. Maintainability testing
2. Test oracle
3. Impact analysis
4. Operational environment
5. Maintenance testing

**Targets:**

1. Assessment of the how a component and its test and development documentation are affected by the implementation of a specified change
2. Tests the changes to an operational system or impact of a changed environment to an operational system
3. Helps determine the expected results for a software
4. Determines ease of maintenance of the system
5. Comprises hardware and software products installed at users' site

Answer

*Impact analysis is the assessment of change to the layers of development and test documentation, and to a component, in order to implement a given change to specified requirements.*

*Maintenance testing tests a system that has been modified but is in operation in a live environment. It also tests the impact on a system of a change made to its operating environment.*

*A test oracle provides a source determining the expected results for a software being tested for comparison with the actual results later.*

*Maintainability testing determines how quickly a system can be modified to meet new requirements, adapt to a changed environment, or correct existing defects, and make future maintenance easier.*

*The software in the operational environment can include database management systems, operating systems, or such other applications.*

**Correct answer(s):**

Target 1 = Option C

Target 2 = Option E

Target 3 = Option B

Target 4 = Option A

Target 5 = Option D

**2. Summary**

Maintenance testing is carried out to test the functioning of an operational system that has been modified. It also tests the impact of a changed environment on a system. The reasons for maintenance testing include maintenance fixes, modifications, or migrations.   
  
Maintenance testing involves all test levels and can also involve any test type. Two of the more common test types that occur during maintenance testing are confirmation and regression tests. Confirmation tests verify the results of the fixed defect. And regression tests retest the entire system again in case the fixes have affected any existing functionality.  
  
To reduce time spent on regression testing, an impact analysis is carried out. The impact analysis determines areas that specifically require regression testing after the maintenance work.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#top)

Models of Software Testing and Development

Learning Objective

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

* *evaluate test types*

**1. Exercise Overview**

In this exercise, you are required to evaluate test types.

This involves the following task:

* evaluating functional, non-functional, structural, change-based, and maintenance test types

You are a tester in a software development company. You are currently involved in two projects. In the first project, you are testing a software order/cash management system. In the second project, you are involved in testing an online audio tool.

Note

*You are currently involved in two projects. In the first project, you are testing a software order/cash management system.*

**2. Evaluating test types**

You are testing a sales order management system for a fast food restaurant. You want to evaluate which test types would be most effective at each stage of the product's life cycle.

Question

The product is at the initial phase of the development life cycle. And the code for the order/cash management system has different elements such as the part that allows for ordering, the part that calculates the prices/taxes, and the part that transmits the order to the cooks.   
  
Each element of the code is developed separately and then sent for testing. The elements will all be integrated at a later date. Which test types would be most effective at this stage of the product's life cycle?

**Options:**

1. Functional testing
2. Confirmation testing
3. Structural testing
4. Regression testing

Answer

***Option 1:****Correct. During the first phase of the product's development life cycle, you check if the product meets the functional specifications.*

***Option 2:****Incorrect. Confirmation testing is conducted on elements that have previously failed a test, and have been submitted for a re-test.*

***Option 3:****Correct. During the first phase, each developed component is subjected to an analysis of the component's internal structure and its implemented behavior. And to do this, you perform structural testing.*

***Option 4:****Incorrect. Regression testing is performed at the end of the product's development life cycle.*

**Correct answer(s):**

1. Functional testing  
3. Structural testing

Question

The elements of code are all now integrated and the complete product is now sent for testing. What test type would be most effective at this stage of the product's life cycle?

**Options:**

1. Confirmation testing
2. Non-functional testing
3. Regression testing
4. Functional Testing

Answer

***Option 1:****Incorrect. Confirmation testing is carried out after the defects found in an earlier round of testing are fixed. Confirmation testing is like a verification round of testing.*

***Option 2:****Correct. As part of system testing you should test that the system has achieved an acceptable level of quality.*

***Option 3:****Incorrect. Regression testing is performed after confirmation testing when the defects that were found are fixed.*

***Option 4:****Correct. Functional testing is conducted to ensure that the integrated system performs according to specification.*

**Correct answer(s):**

2. Non-functional testing  
4. Functional Testing

Question

A number of defects were discovered during the non-functional testing. All these defects were rectified. The product was also passed through a round of confirmation testing. However, there is a slight possibility that the defect fixes altered the overall functionality of the software. What test type would be most effective at this stage of the product's life cycle?

**Options:**

1. Functional testing
2. Maintenance testing
3. Regression testing

Answer

***Option 1:****Incorrect. Functional testing is performed in the initial stages of the product development when you want to check the functionality of any product and verify how it interoperates with other systems.*

***Option 2:****Incorrect. Maintenance testing is performed only after the product is completed and deployed to the customer.*

***Option 3:****Correct. Regression testing is performed when as part of a process, you find or suspect that because of the addition of a new feature, or fixing a bug, a new version of the software may have lost some functionality that was previously present.*

**Correct answer(s):**

3. Regression testing

Question

After regression testing was performed, and the product cleared for delivery, the system was released to the client. A couple of years later, an online ordering/delivery system was added. What test type would be most effective at this stage of the product's life cycle?

**Options:**

1. Non-functional testing
2. Structural testing
3. Maintenance testing

Answer

***Option 1:****Incorrect. Although some non-functional testing may be involved, maintenance testing is called for at this stage of the product’s life-cycle.*

***Option 2:****Incorrect. Structural testing is performed during the various stages of the product's development life cycle and not after the product deployment.*

***Option 3:****Correct. Maintenance testing is performed when you want to check a system operating in a live environment after post-delivery changes are made to it.*

**Correct answer(s):**

3. Maintenance testing

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a02_it_enus/output/html/course_transcript.html#top)

© 2016 Skillsoft Ireland Limited