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

**Course Transcript**

Static Techniques and Test Design in Software Testing

**Static Software Testing Techniques**

| [1. Static Techniques and the Software Test Process](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#t2) |

| [2. The Review Process in Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#t5) |

| [3. Static Analysis in Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#t8) |

| [4. Static Software Testing and Reviews](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#t11) |

**Development and Categories of Software Test Design**

| [1. The Software Test Development Process](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#t15) |

| [2. Test Specifications and Test Cases](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#t26) |

| [3. Dynamic Software Test Design Techniques](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#t18) |

Static Techniques and the Software Test Process

Learning Objectives

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

* *recognize how static techniques can be used in software testing*
* *recognize the difference between static and dynamic software testing techniques*

**1. Static and dynamic testing techniques**

During the software testing process, you can employ different techniques of testing to isolate quality and product-related defects. These techniques largely depend on the manner in which testing is done and the point in time during the product life cycle when they are undertaken.

In the initial stages of development, testing helps you check whether a product meets the necessary requirements and standards. Testing in the later stages is done to check whether defects occur during product execution.

Depending on when the software is executed, testing is categorized as

**static**

In static testing, testing is done before the software is executed. You can use this type of testing to detect defects in the early stages of software development. Typically, in the initial stages of the product development life cycle, all that exists are documents and source code that lay out the design. These are the elements that are tested in static testing with the aim of detecting defects in code and ambiguities in software documents that can cause defects during product development.

**dynamic**

Dynamic testing involves executing the software product while testing. The goal of dynamic testing is to detect runtime defects and analyze the output. You can examine how code and variables behave according to specifications.

Static testing techniques include *reviews* and *static analysis*.

Reviews can test any software work product and are performed manually. This may include software documents such as requirements and specifications. Their aim is to detect defects as early as possible and correct them at the initial stage to avoid issues later.  
  
For example, if a software product does not conform to specified standards, defects may occur when the software is implemented. A review checks the software against these standards so that if defects exist, they can be corrected.

Static analysis involves reviewing source code and the use of automated tools that reduce the time, labor, and cost of going through each code snippet. Automated tools help detect violations in programming standards and code syntax.  
  
Static analysis tests software source code before the system is built and helps avoid waiting until later in the software development life cycle to check if the source code works, which is time consuming and expensive.

Question

You need to develop an Open Office Writer application for which source code has been put in place. You must test the code before the software is executed. Which testing technique is appropriate in this case?

**Options:**

1. Dynamic testing
2. Manual testing
3. Static analysis

Answer

***Option 1:****Incorrect. Dynamic testing occurs on code that is executed and is not done in the initial stages.*

***Option 2:****Incorrect. Manually testing source code is expensive and can also be error prone.*

***Option 3:****Correct. Static analysis provides you with tools to detect defects in source code without the execution of the software.*

**Correct answer(s):**

3. Static analysis

Unlike static testing, dynamic testing involves the execution of the software. This software is run to verify that it meets the requirements and matches the exit criteria stated in the test plans. This testing occurs only after the software is developed so it is applied only to software code and other executable programs.

Dynamic testing techniques primarily validate components and code. These techniques have three broad divisions.

**Specification-based techniques**

The specification-based techniques consider software as a black-box with just inputs and outputs and no knowledge about the components inside. These techniques test what the software does and not how it does it. You can use specification-based techniques when you analyze the requirements or specification of specific software.

**Structure-based techniques**

You can gain knowledge about how the software is implemented and how it works through structure-based, or white-box, techniques. You can use these techniques in component integration testing especially for code coverage.

**Experience-based techniques**

You acquire knowledge and skills in software testing based on your experience, which becomes the basis for experience-based techniques. Using this experience, you can analyze a software design process. When adequate specification is not available about a software product, experience-based techniques are helpful as you have prior knowledge and you are not required to know the specifications. These techniques are highly useful under extreme time constraints as no time is spent on reference while testing the software.

Static testing and dynamic testing differ from one another on various facets:

* static testing is performed in the early stages of the software development life cycle, while dynamic testing is performed during runtime
* static testing is applicable to any software product whereas dynamic testing is only applicable to executable programs
* static testing offers more financial and scheduling benefits compared to dynamic testing
* adherence to coding standards can be checked during static testing, but cannot be checked during dynamic testing

Static testing detects defects in the initial phases of software development before any product is created. Therefore, static testing can be applied to any product such as test cases and documents, whereas dynamic testing is applied only to software that can be executed such as compiled code.

Because static testing is done early in the development phase, any defects revealed at this phase are therefore avoided later on. This saves time and money by reducing the amount of effort required for reworking.

Static testing checks code against specified standards before software is executed in contrast to dynamic testing. For example, you design code and develop it. Before running the software, you verify that your code adheres to coding standards by using static testing. Performing this type of verification during or after the software has executed as done by dynamic testing, results in time delays and increased effort. For example, a programmer working on a large block of code realizes that it is colored as if it is a comment. Then a defect is identified as the improper closing of an earlier comment. The programmer has to again close the earlier comment and then continue working on the code. Such defects can be detected much earlier by using static testing.

Question

Match the types of software testing used to the example scenarios. You can use each type of testing more than once.

**Options:**

1. Static
2. Dynamic

**Targets:**

1. A tester executes a functioning program to verify if outcomes are as expected
2. A programmer runs a functioning program to check statement or code coverage
3. Check and verify security requirements listed in a document for an e-mail client application
4. Prior to code development, a programmer finds that a source code segment is highlighted and needs fixing

Answer

*Dynamic testing is applied only to executable programs and the outcome is compared with the expected output.*

*Code coverage determines the extent to which a source code is tested in a software program. This is identified by dynamic testing in a runtime environment.*

*Requirement documents should be tested during static testing through reviews to highlight defects prior to product development.*

*Source code for software is usually documented. Static testing is applied to identify defects in the code before development.*

**Correct answer(s):**

Target 1 = Option B

Target 2 = Option B

Target 3 = Option A

Target 4 = Option A

**2. Summary**

Software testing techniques include two methods – static and dynamic. Static testing detects defects in the early stages of software development, thereby, cutting down cost and time. It includes two techniques – reviews and static analysis. Dynamic testing is used only for executable programs.  
  
Static testing is highly productive and beneficial. You can apply static testing to all software work products including code. Unlike dynamic testing, static analysis tools can test coding standards and code metrics.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#top)

The Review Process in Software Testing

Learning Objectives

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

* *recognize the differences between the types of reviews in software testing*
* *identify the phases and roles in a formal review process*

**1. Review types and the review process**

One of the most important static software testing techniques is the review. Reviews are particularly valuable because they are executed on documents – including functional and requirement specification documents, which are the foundation for the actual development and execution of the software. As a result, they help find and alert you to any ambiguities, deviations from specified standards, and other defects in documents that – if not corrected – can lead to costly system or product failures later.  
  
In addition to finding defects, reviews also enhance your knowledge about the software product.

There are four types of reviews that you can perform during static testing.

Graphic

*The four review types include Informal, Walkthrough, Technical, and Inspection.*

**Informal**

An informal review occurs when the author of a document requests a peer or technical lead in the same domain to take a cursory glance at the document to validate it. This review does not follow any process though the person doing this review may opt to document findings. Due to its nature, this is a very cost effective.

**Walkthrough**

A walkthrough is undertaken to obtain a common understanding of the document under review. Such a review is led by the author of a document who can answer and clarify doubts that participants may have with regard to document content. Its main purpose is to seek consensus and fresh perspectives from a diverse team in addition to using their knowledge to detect defects in related documents.

**Technical**

Technical reviews are more process oriented than walkthroughs or informal reviews and the findings are always reported. Such a review involves a review team made up of technical experts. The main purpose of this review is to ensure that technical concepts, models, and technical standards identified for product development are accurate and valid for the product. Typically, such reviews are undertaken for products of a critical nature.

**Inspection**

An inspection follows a defined process with clear roles and responsibilities assigned to review team members and is led by a trained moderator. Team members are usually peers and are not associated with the document under review. Similarly, the moderator is also not associated with the product or the document but has the expertise to facilitate the review process. Usually, a person responsible for planning a review leads an inspection team. Using rules and checklists, the inspection team finds and records all defects in an inspection report. They also track whether corrective measures on the document were undertaken and closed. Only when this is done is the inspection review considered complete.

You may choose to use only one review type or use all depending on the criticality or nature of the software product.   
  
All four review types vary from one another in terms of formality with some categorized as more formal than others.

Graphic

*Description of the Level of formality in review types diagram:  
  
On the spectrum of formality, the lowest level of formality is characterized by the informal review. On the opposite end, the most formal review is inspection. Walkthrough and technical reviews are in the middle with walkthrough being less formal than technical but more formal than the informal review.  
  
Description ends.*

The degree of formality depends on the organized structure and documentation that a review follows. For example, technical reviews and inspections involve a very formal process, while walkthroughs are less formal.

Depending on the nature of the software product or the business needs of the product, you can employ different review techniques. You can perform an informal review before a technical review. For example, you may need to perform a minor maintenance task on a software application. Because it is just a simple maintenance task, you may simply request a co-worker or a peer to review it. This is an example of an informal review.

But if you have authored a document for a product that affects different teams you may decide to have a diverse group analyze the document and offer feedback and perspectives on it. Because some members may not be familiar with the work process or specifications used in the document, you choose the walkthrough review so you can walk participants through the document.

Suppose a particular software product is highly technical in nature and requires the use of complex technical concepts and coding. In such cases, you may opt for a technical review wherein you ask technical experts to arrive at an agreement and validate the technical content of the product under review. Any defects in the document may easily be identified by such a team and rectified to avoid delays and expenses later.

Similarly, if a product is being developed for a safety critical industry, such as the airline industry, you may opt for an inspection – which is the most formal of reviews. In this instance, reviewers with experience in building similar software and with exposure to the specific industry can help identify deviations or defects in the document to avoid serious issues later. The product under review is checked thoroughly by the reviewers, and defects identified are tracked closely.

The success of a review is dependent on

**defined objectives**

Reviews must have clear, predefined objectives which allow reviewers to choose the proper review type. This helps reviewers focus only on relevant aspects making them work in a streamlined process to achieve the objectives. As a result, their time is utilized efficiently and effectively.

**the nature of the review team**

For a review to be successful, the right people should be involved in the review process. For example, reviewers should have the expertise or knowledge in the product domain so they can locate defects and provide recommendations. Similarly, someone leading the process should have the experience to guide participants.  
  
If participants require training on some of the processes, then that training must be provided for the review to be effective.

**how defects are communicated**

Defects found during a review should be communicated objectively and constructively. The idea is to create a positive experience for the author and the participants so that the main purpose of a review is met without making it a personal attack on the capabilities of the author or the review team.

**the review techniques used**

Another factor that impacts the success of a review is the choice of review techniques used. For example, a software product enhancement may do with just an informal review or a walkthrough but a product that is expected to be groundbreaking and of immense value would require an inspection to ensure that quality and standards are not compromised.

**the review process**

The review process, when it follows a defined plan, ensures that time and costs incurred in reviews are not wasted. Instead, it charts out a clear course of action with adequate time for the reviews and rework incorporated. As a result, it prevents hastily performed reviews that may not uncover defects.

Question

Match the scenarios with the type of reviews appropriate for each.

**Options:**

1. Technical
2. Inspection
3. Informal
4. Walkthrough

**Targets:**

1. Different members of a review team individually spell-check and test different elements of code against specifications
2. A developer presents the material to be reviewed to a group to facilitate discussion and understanding
3. A team of people checking if interface requirements of an application are valid
4. A programmer asks a couple of colleagues to check some code he has just written

Answer

*An inspection involves participants of the review individually checking a software product against standards and specifications. This review requires participants to have domain expertise.*

*In a walkthrough, reviewers gain a common understanding about the material under review from the author.*

*Reviewing interface requirements requires technical expertise and is done by technical reviewers.*

*An informal peer review does not need to follow any process and can be used anytime.*

**Correct answer(s):**

Target 1 = Option B

Target 2 = Option D

Target 3 = Option A

Target 4 = Option C

Regardless of the types of review undertaken, the basic review process necessarily requires certain elements – existence of a document to be reviewed, a person or persons identified to review that document, communication of defects or issues in the reviewed document, and updating the document, if required.  
  
Technical reviews and inspections follow a formal process with defined phases that differ from less formal reviews such as walkthroughs, where phases may or may not be followed explicitly.

The formal review process consists of six stages.

Graphic

*Description of the formal review process:  
  
The stages in a formal review process include planning, kick-off, preparation, review meeting, rework, and follow-up and follow the same sequence as listed.  
  
Description ends.*

**Planning**

Planning involves selecting people to carry out the review process and assigning them responsibilities. During this stage, a review leader must determine who will review what part of the software document. Entry and exit criteria for the review are also defined here.

**Kick-off**

Once planning is over, you distribute the documents to be reviewed to reviewers in the kick-off phase. Usually, this is done as a meeting where review objectives are introduced and time commitments are agreed upon. The goal of a kick-off is to ensure that everyone understands the document under review and its need, and to clarify doubts or ask questions.

**Preparation**

After kick-off is the individual preparation stage, where participants in the review read the source documents, check them against checklists, and identify defects and questions. This is done individually by each member of the review team and each of them records defects, and questions noted separately. A time period is set for this phase so as to complete the review on time.

**Review meeting**

In a review meeting, the review team comes together with the defects and their severity levels found during the preparation stage. At this meeting, they log the defects as per severity levels, and also analyze and discuss the defects that require immediate attention. At times, recommendations and work-arounds are also discussed. Minutes of this meeting are recorded and the defect log is then given to the author.

**Rework**

After the defect log is handed to the author in the review meeting, the rework phase begins. In this phase, the author of a document fixes the defects identified as a priority, ensuring that changes in the document are easily traceable. Issues not fixed should contain proper reasons from the author.

**Follow-up**

The last stage of the review process is when the leader of the review follows up on the corrections made for the logged defects. The leader also collects information such as the number of defects found, the time spent to correct them, and the total effort spent on the review. This information is stored for future analysis, based on which, the leader suggests changes to improve the process.

The formal review process is characterized by defined roles and responsibilities that each member in the review team is assigned with. These roles include

**manager**

It is a manager's responsibility to manage the review process including assigning time for reviews in the project plan. A manager checks if defined review objectives are met. To facilitate this process, managers must decide if any members from the review team need to be trained and accordingly – schedule their training before the review begins.

**moderator**

Decisions regarding the review such as entry and exit criteria, review meetings, and documentation are taken by the moderator. He or she also acts as a facilitator during discussions to ensure that they do not lose focus on the objectives of the review.

**author**

The author develops the document to be reviewed and fixes the errors found during review. The author helps the team understand gray areas and more importantly, should recognize the quality issues that can be avoided in future documents.

**reviewers**

Reviewers check for defects in a document based on the experience and domain knowledge they possess. They follow certain standards and checklists during the review.

**scribe**

All formal meetings require that points raised and discussed are documented. In a review meeting, this is done by a scribe. It is the scribe's responsibility to ensure defects and suggestions are logged clearly so that authors can understand them without confusion.

Question

Match the formal review phases to the activities performed in those phases.

**Options:**

1. Planning
2. Kick-off
3. Review meeting
4. Follow-up

**Targets:**

1. Log defects as per severity levels
2. Distribute review documents to reviewers and discuss review objectives
3. Select members of the review team and assign distinct roles to them
4. Collect information on the total review effort and use it for future process improvements

Answer

*A review meeting is conducted as part of the review. In this stage, the defects identified during the review are logged and recorded according to their severity levels to help in future review processes.*

*The documents to be reviewed are distributed to the reviewers before a review begins. The reviewers discuss these documents and the objectives as part of their review.*

*Planning involves identifying the right people for the review team based on their experience and skills. Each selected member is then assigned a specific role such as reviewer or moderator.*

*Details about the number of defects and the time spent to correct them are gathered and stored during the follow-up stage. This information can help in improving processes in the future.*

**Correct answer(s):**

Target 1 = Option C

Target 2 = Option B

Target 3 = Option A

Target 4 = Option D

Question

In a formal review, you are given the responsibility of facilitating review meetings and ensuring the quality of the review process. You are also required to help define entry and exit criteria for the review. What role are you assigned within the review process?

**Options:**

1. Reviewer
2. Scribe
3. Moderator
4. Author

Answer

***Option 1:****Incorrect. A reviewer checks a document against set standards and checklists and tracks the defects.*

***Option 2:****Incorrect. A scribe documents the defects and the suggestions discussed during review meetings.*

***Option 3:****Correct. A moderator facilitates and leads the review process and leads the review team with an eye on maintaining the quality of the review process.*

***Option 4:****Incorrect. An author is the creator of the document under review and is responsible for rectifying defects raised during review.*

**Correct answer(s):**

3. Moderator

**2. Summary**

The review is the most widely used static software testing technique. There are many types of reviews which differ in levels of formality. They are informal reviews, walkthrough, technical, and inspection.  
  
Formal reviews follow a very organized process with well-defined phases such as planning, kick-off, individual preparation, review meeting, rework, and follow-up. These phases involve various people with specific roles and responsibilities.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#top)

Static Analysis in Software Testing

Learning Objective

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

* *recognize how static analysis could be used in software testing*

**1. Static analysis in software testing**

Static analysis is an important static technique in software testing. It is used to detect code-related defects. Because code is analyzed without it being executed, this is considered a static technique. Static analysis identifies suspicious aspects of code enabling them to be rectified thereby improving the quality and maintainability of code.  
  
Another aspect checked by static analysis is software models identified for the product. These are pictorial representations of how software objects will relate to one another in the final solution. Analyzing these before they are implemented can help alert you to any gaps or logical errors in the way objects are shown to be related.

Static Analysis is performed in the early stages of development to detect defects such as breach of standards and metrics. Because it occurs earlier in the product development life cycle, it provides early detection of the defect density in the software. Consequently, it reduces the cost and time taken to rectify them at a later stage in development. Static Analysis also helps reduce the number of defects found during dynamic testing.

Graphic

*Static Analysis occurs in the Analysis and Design stage and Code Generation phase of the Product development life cycle.*

Analyzing source code and software models manually is time consuming and laborious. In addition, it is also prone to errors as manual checking is open to misinterpretations and keying errors, for example. Therefore, static analysis is most effectively performed using automated tools that inspect elements of the code and the software model to check if defects exist in the logic or code syntax. Certain compilers could be considered to be tools to check for defects in code since many of them do report syntax errors and other defects.

Static analysis tools can detect critical defects that exist in the code or in software models in the form of

**undefined values in variables**

You should give specific values to variables because undefined variables can cause erroneous calculations. Variables should also not be left unused. Declaring a variable and not using it in a program may cause a part of the program to be omitted.

**unreachable or dead code**

Unreachable or dead code is not logically included in a program and so it cannot be executed. It also remains untested throughout the product development life cycle and may cause a program to fail during runtime.

**variation errors in interface**

Inconsistent interface within and between software components may be due to a conflict of shared objects between modules. For example, module A requests two values from module B, which has only one output. Therefore, a mismatch of values sought and output received prevents these modules from being integrated.

**nonadherence to standards**

Violations or non compliance with coding standards results in programming standards violations. Incorrect use of a software language may cause syntax violations in code and software models.

**code vulnerability**

Security issues may arise when source code is left unsecured or vulnerable. User credentials that are not secure may also cause security problems.

**programming language errors**

Syntax violations in the code that are detected by static analysis tools when compiling a program.

Question

What types of defects faults do static analysis tools detect?

**Options:**

1. Code variables without a value
2. Code failures at runtime
3. Errors In programming
4. Ambiguous product specifications

Answer

***Option 1:****Correct. Undeclared or undefined variables cause failures when code is executed and are detected by static analysis tools.*

***Option 2:****Incorrect. Runtime failures are identified by dynamic testing, not by static analysis.*

***Option 3:****Correct. Failures that arise when code is compiled are detected using software compilers during static analysis.*

***Option 4:****Incorrect. Ambiguities in specifications are checked through review techniques of static testing and not by static analysis tools.*

**Correct answer(s):**

1. Code variables without a value  
3. Errors In programming

You can use static analysis tools to check whether a software product adheres to coding standards. Checking coding standards – code naming conventions or the manner in which programming rules are applied, for example – is critical because non-adherence to them affects the readability and quality of your code. Static analysis tools help detect such problems faster, as they are automated and programmed to look for such issues.

Code metrics or attributes of code structure is another aspect of code that static analysis focuses on.  
  
Many times, structures in source code can become lengthy and complex such as code with many lines, decision points, branches, and loops. The larger and more complicated the code is, the more difficult it is to understand and implement, thereby leading to errors. Such code is also more difficult to maintain.

Code metrics measure the structural complexity of code and can be used to indicate potentially risky problem areas.

You can also use static analysis to check the control flow of a software program. Control flow describes the logical structure of a software program and is represented as a diagram showing linear paths in a program.  
  
Analyzing control flow in the early stages of development provides the program execution sequence and helps identify code that does not fit in. As a result, unwanted or unused code is avoided.

Question

Identify the scenarios where using static analysis to test is appropriate.

**Options:**

1. Two developed software modules must be checked for integration defects
2. Code fragments in the source code seem to have high code complexity
3. The source code is ready to be executed and ready to be checked for system or program failures
4. Source code is checked for comments describing rules and to check the flow control

Answer

***Option 1:****Incorrect. Integration testing tests the functional requirements and whether the results are as expected. This is done during dynamic testing and not checked using static analysis.*

***Option 2:****Correct. Static analysis tools identify the code complexity and detect hidden errors in code.*

***Option 3:****Incorrect. Checking for system failures in a software program which is ready to be executed is performed through dynamic testing.*

***Option 4:****Correct. Source code is checked for comments and for flow control using static analysis tools.*

**Correct answer(s):**

2. Code fragments in the source code seem to have high code complexity   
4. Source code is checked for comments describing rules and to check the flow control

**2. Summary**

Static analysis is an important static testing technique which provides tools to detect programming defects before run time. They are best used immediately after code has been written and during software modeling. They can detect logical and security errors in an application. These tools analyze source code and check for adherence to code metrics. Code metrics help you understand the complexity of code and help maintain the quality of a software program.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#top)

Static Software Testing and Reviews

Learning Objectives

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

* *recognize how defects can be found with static software techniques*
* *recognize the benefits of different types of reviews in software testing*

**1. Exercise overview**

In this exercise, you want to identify defects that may be found using static testing techniques, and also recognize how different types of reviews in static testing helps.

This involves the following tasks:

* identify the type of defects found using static testing techniques
* determine the benefits of the different types of reviews

You are a software developer. You want to develop software for a digital media player. A team has been set up to develop this product. This team has collated requirements and created documents to support the development of the product. In order to be competitive, the software must meet industry standards. This has been researched and documented by the team. Source code will adhere to industry standards and requirements.  
  
The team is in a position to begin developing this product now.

**2. Using static testing techniques**

Before developing the software product for the digital media player, you decide to apply static testing techniques on the work done so far.

Question

What types of defects are likely to be found using static testing techniques on an audio software product that is to be developed?

**Options:**

1. Chips used do not support a specific bitrate for compressed audio
2. Sound quality using the audio software on the digital media player is poor
3. Interfaces and links in the identified software model are incorrect
4. Audio software seems incompatible with certain brands of digital media player

Answer

***Option 1:****Correct. Chips that supporting the bitrate for the audio are a requirement. Requirements of the software are inspected before development through static testing.*

***Option 2:****Incorrect. Checking sound quality can be done once the audio software is developed and is an instance of a runtime software failure. Such defects are identified during dynamic testing.*

***Option 3:****Correct. Interface specifications not matching the design of the software model are identified before the software is developed. Static testing identifies such defects.*

***Option 4:****Incorrect. To check compatibility of the software product across brands requires that the product be compiled and executed on the brands. This is done through dynamic testing.*

**Correct answer(s):**

1. Chips used do not support a specific bitrate for compressed audio   
3. Interfaces and links in the identified software model are incorrect

Question

What type of errors do you identify using static testing in the audio software you develop?

**Options:**

1. Ensuring that the end products runs as intended
2. Lines of code
3. Code complexity

Answer

***Option 1:****Incorrect. Ensuring that the software executes correctly is done by dynamic testing.*

***Option 2:****Incorrect. Calculating the lines of code enables you make sure code adheres to any size standards and aids in its maintainability. It does not help you identify complexity errors.*

***Option 3:****Correct. Complexity of code is calculated using the code complexity metric during static testing.*

**Correct answer(s):**

3. Code complexity

**3. Determining benefits of reviews**

Having identified the types of defects you are keen to detect using static testing, you can now employ different types of techniques available to perform static testing.

Question

For the audio software being developed, another organization has been tasked with creating the software that will help your product interface with the digital media player. To ensure there are no defects that turn up at a later stage, you want to ensure that an independent team led by a trained moderator reviews all documentation prepared by the other organization thoroughly. Which types of review are most appropriate in this scenario?

**Options:**

1. Code analysis
2. Inspection
3. Walkthrough
4. Technical

Answer

***Option 1:****Incorrect. Testing software code at this stage is performed using static code analysis while software documents must be reviewed.*

***Option 2:****Correct. The high-level software documents such as requirements and design specifications are inspected thoroughly by a team of experts.*

***Option 3:****Incorrect. Walkthroughs can be conducted as either formal or informal reviews and are typically led by the authors of the software.*

***Option 4:****Correct. Testing interfaces include troubleshooting technical inconsistencies in software. Such inconsistencies are usually identified during technical reviews.*

**Correct answer(s):**

2. Inspection  
4. Technical

Question

You also want to perform static analysis on source code and associated software models received from the organization developing the software. What benefits would the static analysis technique provide?

**Options:**

1. Allows changes to be made to complex code prior to development
2. Identifies complex, high risk areas of code
3. Timely detection of incorrectly interpreted specifications
4. Helps determine whether documents conform to documentation standards

Answer

***Option 1:****Correct. Static analysis tools allow you to reduce the complexity of the code before development, which prevents costly failures at a later stage.*

***Option 2:****Correct. You can identify complex, and thus high risk areas of code.*

***Option 3:****Incorrect. Software specification documents are tested and analyzed using review techniques during static testing and not by static analysis.*

***Option 4:****Incorrect. Compliance with software documentation standards is tested using reviews during static testing.*

**Correct answer(s):**

1. Allows changes to be made to complex code prior to development  
2. Identifies complex, high risk areas of code

You recognize how defects may be found using static testing techniques. You recognize the benefits of different types of reviews in software testing.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#top)

The Software Test Development Process

Learning Objectives

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

* *recognize key terms used in software test design*
* *recognize the components of a software test case*
* *recognize how a software test procedure specification is designed*

**1. Keywords in software test design**

During the software testing process, you can use both static and dynamic testing techniques. Static testing techniques involve review and analysis of specifications and code to detect defects in design before the code is executed or implemented. Dynamic testing is where you actually run the developed code to check for failures at runtime.

The software test development process essentially covers dynamic testing techniques because this type of testing requires you to design tests. To test code for runtime failures as performed in dynamic testing, you need to identify test conditions, specify test cases, and create test procedures to use.

Some of the keywords you are likely to encounter during test design are

**test condition**

A critical element in any test is to decide the focus of the test, which then becomes the test condition. Depending on the software tested, a test condition specifies the characteristic, an item, or an event of the software that you want to test. For example, in a user authentication system, test conditions may include checking if the system accepts alphanumeric passwords that are six characters long and a user name which is a string of letters.  
  
Although you can have as many test conditions as you want, checking all may not be feasible. So you filter them to identify only the important conditions.

**test case**

To test one or more test conditions, you need test cases. A test case is specific to the condition being tested, and will use elements such as input values, preconditions for test execution, and expected results to check if the software performs as designed.  
  
For example, in a user authentication system, you may want to create a test case to check the password validation function. The test case will clearly state the range of values allowed, the scope of the test, and the expected outcome for each set of input values. The software is then tested against these specific values and outcomes.

**test procedure**

The procedure to follow while implementing the test case to verify test conditions is referred to as the test procedure or test script. It details the steps to be performed and the order in which they need to be executed.  
  
For example, the test procedure to check password validation lists the steps to implement the test. You might first test if it accepts six characters and then check if they are alphanumeric.

Depending on the level of formality of testing followed in an organization, documents that adhere to the IEEE 829 Standard may be used to capture test conditions, test cases, and test procedures. Documentation helps create a well-controlled and unambiguous testing process because information about what is to be tested and the expected results are clearly stated.

Note

*IEEE 829 Standard is the Test Documentation Standard.*

The IEEE 829 Standard document called *Test Design Specification* serves as a template for documenting and communicating test conditions during the various testing stages. It also documents the test approach and the exit criteria for the test condition.

Specifics of how to verify test conditions are detailed in the IEEE 829 Standard document called *Test Case Specification*. This document can record one or more test cases, and typically, lists inputs for testing, expected behavior or results, dependencies, and the testing environment.

After creating the test cases, you logically group and order them by specifying the test procedure or the sequence of component steps. This is documented in a *Test Procedure Specification* document – an IEEE 829 Standard document.   
  
This document helps testers follow the test procedure by implementing tests in the order in which they are prioritized to achieve the desired result.

Question

Match the key software test design terms to their descriptions. One keyword option will not be used.

**Options:**

1. Test condition
2. Test case
3. Test procedure
4. Test design

**Targets:**

1. Tests one or more test conditions using specifics such as input values and expected results
2. Details how the test case will be executed or implemented
3. Specifies the characteristic of the software you want to test

Answer

*A test case uses specific details to test one or more test conditions to check if the software performs as expected.*

*A test procedure identifies the list of steps and the sequence in which steps should be followed to implement a test case. It also specifies the order of execution of test cases.*

*A test condition is the characteristic of the software you want to test.*

**Correct answer(s):**

Target 1 = Option B

Target 2 = Option C

Target 3 = Option A

**2. Components of a software test case**

Test conditions are identified during the test analysis stage by studying the software requirements and defining tests to match the requirements. You create test conditions based on test data, test inputs, and test outcomes.

Note

*Test data refers to the data residing in the system under test. The test data can be modified during test execution.*

For example, say you are developing an order processing system. The test data here includes item and order details stored in the database. And you can create test conditions to check if the stock position changes appropriately when orders are placed or if a new record is added with appropriate details for each new order.   
  
Test inputs for this example include the details entered in an ordering form. You create test conditions to check if these inputs are correct and as expected by the system. You also create test conditions for checking the test outcome, for example, if a message is shown to the user when incorrect input is provided.

When listing test conditions, you first include all potential items that can be tested. You then identify the critical ones to be retained as test conditions and prioritize them.

You need to ensure the test conditions you choose encompass all possible areas or components to be tested. In addition, ensure the test conditions can detect most of the defects in the software.

During the test design stage, you design one or more test cases to check the test conditions. You can also combine related test conditions into one or more test cases.

For example, assume you are testing a user authentication system. You create test conditions for each field on the screen, organize them into test cases, and create a test procedure specifying the order of execution for the test.   
  
If the username field accepts a maximum of 30 alphanumeric characters and will only allow the special underscore '\_' character in addition to the upper and lowercase letters, you can create test conditions to check these.

Both test conditions and test cases need to be traceable. *Traceability* refers to being able to track the test condition or test case to their sources in the requirements and software.  
  
Ensuring traceability helps to find a fix for traceable test conditions when project requirements change or a set of routine tests start malfunctioning.  
  
Say the client changes requirements for the username field. The traceability characteristic helps to modify the test cases and conditions that check this field.

You can document identified test conditions in the Test Design Specification document using the components available in the template.

**Test design specification identifier**

The test design specification identifier is a unique number used to identify the document. Usually, the naming convention used for the software is used to create this identifier to aid easy identification. In addition, you can provide other details – a short name, version name and number, author name, and revision history – in this component.

**Features to be tested**

The features to be tested section is where you list the set of objectives covered by the test conditions, and where you mention the test level appropriate for the test item along with the source documentation reference. Test objectives define the aspects or features of the software to be tested.

**Approach refinements**

Any refinements to the test approach or test design from what is listed in the test plan is detailed in the approach refinements section. Information such as the test techniques, the selection criteria for test techniques, methods for result analysis, and testing levels for test items are detailed here.

**Test identification**

You specify the details of the individual test cases – level of testing and the test technique, for example – and the test procedure mapping to them in the test identification section of the template.

**Features pass/fail criteria**

Testers assess if the test failed or passed using information you enter in the features pass/fail criteria section. In this section, you tabulate each feature tested, the test case ID to which it maps, and its pass or fail criteria.

Question

Match the components of the test design specification template with their explanation.

**Options:**

1. Features to be tested
2. Approach refinements
3. Test identification
4. Features pass/fail criteria

**Targets:**

1. Specifies the conditions that enable you to check the outcome of the test and decide if it succeeded
2. Specifies details of the test cases and test procedures mapping to the listed test conditions
3. Details test objectives covered by the test conditions
4. Lists the selection criteria for the test techniques

Answer

*The features pass/fail criteria component is used to specify the criteria for deciding whether a feature has passed or failed.*

*The test identification component is used to trace the test conditions to the related test cases and test procedures.*

*You use this component to trace the test condition to check the feature or component of the system it targets.*

*The approach refinements component is used to specify the test technique selection criteria and applied result analysis methods.*

**Correct answer(s):**

Target 1 = Option D

Target 2 = Option C

Target 3 = Option A

Target 4 = Option B

Test conditions give a generic description of what is to be tested. However, prior to starting the actual test, you need specific details, which even a novice tester can use to run the test and clearly state whether the test succeeded or failed. If exact input values and their specific results are given, testers cannot misinterpret the test results.  
  
You make the test conditions more specific while creating test cases by providing the exact input values. You can also cover one or more related test conditions in a test case. Test cases are created along with test data during the test design phase of the software testing process.

The actions you use to select the test cases are called test design techniques. When you apply these techniques, you select a subset of all tests that detects all possible types of defects in the system. A good test case is characterized by traceability and having clear input and output specifications.

A test is useful only if it checks the results of the software. This requires knowledge of the expected outcome of the test case to know if it succeeded or failed. A *test oracle* can serve as the base for obtaining information about expected results, thus helping you cross-check the actual test results of the software.  
  
A requirements specification document, a user manual, or expert comments can serve as test oracles, as they provide a basis for recognizing expectations of the software.

To create complete test cases, you should provide all the details required by the seven components of the Test Case Specification template.

**Test case ID**

You use the test case specification identifier to specify a unique number to identify the document. You can base this identifier on the software component or features to which the test case pertains. You can also provide details such as the test case version name and number, version history, or the author name and contact information.

**Test items**

The test items component lists all the components, features, and items to be tested. You can also include reference to the source document to ensure traceability.

**Input specifications**

You use the input specifications component to list the inputs – for example, test data elements, values, user actions required – to implement the test case. Testers execute the test case by feeding in these input values to check the behavior of the software or features being tested. Depending on the level of testing, you can also include inputs such as tables, conditions, and relationships.

**Output specifications**

For every input specified, there will be an expected output that needs to be specified in the output specifications component. This section with expected behaviors is essential to confirm if the test succeeded or failed.

**Environmental needs**

The environmental needs component describes the hardware and software platforms required for testing. These should be similar to the real environment for the software for testing to succeed. Details of the operating system, tools, applications, compilers, and configurations for software use are included here.

**Special procedural requirements**

If software components have special requirements or constraints to be considered for test case execution, you specify them in the special procedural requirements section. These could include details such as when to schedule the tests, load, setup requirements to test a specific aspect, output location, identification and operator intervention.

**Intercase dependencies**

If a test case is dependent on any other test cases – either as a prerequisite to other test cases or is a follow-on test case for other tests – you specify them in the intercase dependencies component. This ensures traceability.

After test cases are created, you can prioritize them so that critical test cases are executed first. This helps you to decide which tests to ignore and which are the least important ones to be executed.

Question

The Password field in a user authentication system needs to be tested. Choose the best test case from those specified in the learning aid [Test cases for User authentication system](javascript:doWindow('./lasd_sftf_a03_it_enust103_frame.html'))*.*

**Options:**

1. Sample A
2. Sample B
3. Sample C
4. Sample D

Answer

***Option 1:****Incorrect. This sample test case lacks important details. However, you can use this test case if output specifications are added.*

***Option 2:****Incorrect. Although this test case contains important details, it provides actual results instead of the output specifications, which are required to check if the test succeeded.*

***Option 3:****Incorrect. This test case does not contain details about the test items being tested. As a result, traceability becomes an issue.*

***Option 4:****Correct. This test case contains all the required details needed to make a test case as specific as possible.*

**Correct answer(s):**

4. Sample D

**3. Software test procedure specification**

In some projects, dependencies of features force you to execute the test cases in a specific order because only then is the test considered valid. For example, in the user authentication system, you need to validate the Username field, check if the control is passed to the Password field when the user presses Enter, and then validate the password specified.   
  
In the test implementation stage, you create a test procedure specification document that specifies this order of execution. The test procedure specification is also called a test script.

Note

*Test scripts created for tests planned to be run manually are called manual test scripts.*

You also group related test cases into a test procedure. This is done to group related test cases pertaining to a specific software feature or specification element and execute them as a block. For instance, in the user authentication system, you might group all the test cases related to the username field.

Testing is done within time constraints. So test designers often follow the *Find the scary stuff first* principle to prioritize tests.  
  
According to this principle, you give high priority to test cases that check the basic features or items. These are potential threats to the functioning of the software if not tested.

You can detail the test procedures using components of the Test Procedure Specification template.

**Test procedure specification identifier**

A unique number generated for each document based on the software naming convention is used as the test procedure specification identifier and should also indicate the level of the test and the software. This component can also include details such as version name and number, the version author, and version history.

**Purpose**

You use the purpose component of the template to describe the procedure to be followed and to mention the test cases to which it applies.

**Special requirements**

Any facets of the test procedure – such as whether the testing is manual or automated, if there are any procedures that act as a prerequisite to this, or if the test environment is similar to the production environment – are detailed in the special requirements component of the template.

**Procedure steps**

The procedure steps component lays out specific activities that a tester needs to perform in order to execute the procedure. These could include sequences of steps for specific activities such as logging in to the software, setting up the software, or navigating through the software.

For example, you want to create a test procedure for a user authentication system that targets the fields in the authentication screen. To do this, you sequence the set of activities for the test procedure:

* first check if the cursor is in the Username field
* type Jonathan\_Gold in the Username field and press Tab
* check if the cursor moves to the Password field
* type allow89 in the Password field and press Tab
* check if the control passes to the Login button
* click the Login button
* check if the Welcome user screen is displayed

Question

A test procedure lists the basic set of tasks to test a spreadsheet application – entering data, calculating the sum, and then saving the result. Arrange the options in the correct order and then select done.

**Options:**

1. Create a blank spreadsheet
2. Enter the data for row 1: 1, 2, 3
3. Enter the data for row 2: 5, 4, 3
4. Enter the data for row 3: sum(a1, b1), sum(a2, b3), sum(a3, b3)
5. Save the file as "testspreadsheet1"
6. Close the spreadsheet

Answer

**Correct answer(s):**

**Create a blank spreadsheet is ranked**

The first basic task in testing a spreadsheet application is creating a blank spreadsheet.

**Enter the data for row 1: 1, 2, 3 is ranked**

After creating a blank spreadsheet, the next task is entering data in a row.

**Enter the data for row 2: 5, 4, 3 is ranked**

If you want to check the sum function, you need to add another row of data.

**Enter the data for row 3: sum(a1, b1), sum(a2, b3), sum(a3, b3) is ranked**

After specifying the data in two rows, you need to specify the data in the third row and use the sum function.

**Save the file as "testspreadsheet1" is ranked**

After entering data and using the basic sum function, you can save the spreadsheet.

**Close the spreadsheet is ranked**

The final basic task involved in testing a spreadsheet application is closing the spreadsheet.

**4. Summary**

The software test development process focuses on dynamic testing that involves the actual test execution of the code. The test design process begins with identifying the software component or feature that you want to test – the test condition.  
  
Test conditions are documented in a test design specification template. Test cases are then created – using the test case specification template – to test one or more test conditions. Various test cases are then grouped based on dependencies and priorities and ordered to create a test procedure. Such test procedures are documented using a test procedure specification template.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#top)

Test Specifications and Test Cases

Learning Objective

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

* *evaluate test case quality and create test procedures from test cases*

**1. Exercise Overview**

**2. Explaining different test specifications**

**3. Evaluating the test case**

**4. Creating test procedures**

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#top)

Dynamic Software Test Design Techniques

Learning Objective

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

* *recognize when to use the different dynamic software testing techniques*

**1. Choosing a dynamic test design technique**

An integral part of any software testing is to check whether the code runs as expected when executed. This is undertaken through dynamic testing, where the code is executed to check for defects.   
  
Three different approaches are available within dynamic testing to design tests for the code. Using these techniques you create test conditions and test cases.

The three approaches available in dynamic test design include

**specification-based techniques**

Specification-based techniques are also called black-box techniques. These techniques check the outcomes of the software by feeding inputs into the system and testing if they are achieved by checking the output received. They can also be used to derive test conditions and cases from the functionality of the system.

**structure-based techniques**

Techniques that check how the software or system functions internally are called structure-based techniques. These techniques are also called white-box techniques.

**experience-based techniques**

To create or choose test cases based on the tester's experience of working with similar systems or the general experience of the tester, you can use experience-based techniques.

Specification-based techniques create test cases based on specifications – functional or nonfunctional – of the proposed software or software models. Functional aspects tested include features and functions of the software or system. You can also use black-box testing to check quality-related aspects of the system such as its performance.

Because this category of techniques does not examine the process but only checks if the output obtained is appropriate for the input supplied, the techniques are called black-box testing techniques.

Note

*Black-box testing is also known as behavioral or input/output driven testing.*

You test the software to ensure that it functions as expected. Structure-based testing techniques are used to check the operation of the software or system.

This testing technique is based on the internal structure of the component or system and uses it to create test cases. For example, you can test if the looping, branching, decision, and other constructs function properly.  
  
Because the tester focuses on what happens within the system or how a system processes the input to deliver an output, these techniques are also called *white-box* techniques. The internal structure and the functioning of the code is the focus of the tester.

Note

*Structure-based techniques are also known as glass-box testing, clear-box testing, and non-opaque box testing.*

Experience-based techniques can also be used to harness the knowledge people have gained from previous experiences with similar software and environments to create better test cases.

Note

*Experience-based techniques can include the knowledge of testers, developers, users, and other stakeholders involved in the project.*

Their experience adds a different and additional perspective to testing and can be useful when you need to create test cases quickly using minimal resources. Generally, these techniques are used in combination with structure-based and specification-based techniques. However, for low-risk systems, systems that require short turnaround time, and systems with out-of-date or inadequate specifications, using experience-based techniques might be the only means to create test cases.

Question

Match the characteristics of the testing techniques with the appropriate category of dynamic testing techniques. You may use each category more than once.

**Options:**

1. Performed by testers, developers, users, and other stakeholders
2. Also called black-box testing
3. Tests what the system does
4. Uses the previous knowledge and skills of people to design test cases
5. Identifies how the software or system functions

**Targets:**

1. Specification-based testing techniques
2. Structure-based testing techniques
3. Experience-based testing techniques

Answer

*Specification-based or black-box testing techniques focus on functional and nonfunctional specifications of the software by checking the outcomes of the software by feeding inputs into the system.*

*Structure-based or white-box testing techniques check how a system or a component works by testing the internal structure of the system. Because it requires knowledge of how the internal structure works it is typically performed by developers.*

*Experience-based testing techniques use previous knowledge and experiences of people about the system to derive test conditions and test cases. They are especially useful for low-risk systems, systems that require short turnaround time, and systems with out-of-date or inadequate specifications.*

**Correct answer(s):**

Target 1 = Option B, Option C

Target 2 = Option E

Target 3 = Option D, Option A

You might need to choose one or more dynamic testing techniques to create test conditions and test cases, as they provide a comprehensive check of the software or system components being tested.   
  
This is because different testing techniques detect specific types of design and structural defects. When you use a combination of these techniques, you can detect most of these possible defects.

You can compare the characteristics of the three categories of dynamic testing to choose the techniques you want to use

**black-box testing techniques**

Black-box testing techniques check the functions of software or component. Using these techniques, you isolate defects in the results and behaviors of the software. These techniques are predominantly performed by testers including specification based and experience based techniques.

**white-box testing techniques**

The focus of white-box testing techniques is on testing the structure of the system and its internal functioning. Using these techniques, you can isolate the bugs in the functions, data structures, or interfaces. These techniques are predominantly performed by developers and are typically used once the code has been written.

**experience-based testing techniques**

Techniques that use the previous experiences of people to arrive at the test cases are called experience-based techniques. Using these techniques, you gain knowledge about possible defects and their distribution in the software system, its usage, and environment. These involve testers, users of the system, developers, and all stakeholders. There are no specific tools used to apply these techniques.

Each category of technique is best suited for specific scenarios. If a software or system has clearly defined and detailed formal specifications, you can apply specification-based techniques to create test cases for all levels of testing from component testing to acceptance testing.

For example, you may want to create test cases and test conditions for a specific software that is yet to be developed or for which actual code or structures are not yet created. In this instance, all you have available are formal specification documents that mention what the software is expected to do or how it should behave in a specific condition. You use these specifications to create appropriate test cases and then implement specification-based testing techniques in this scenario.

Structure-based techniques are used when you have tool-support for code or when you have automated unit test regression suites. It is also used at all levels of testing such as system and acceptance testing.

Your decision to choose this type of testing will depend on whether the tester understands code or structures to be tested. If so, you will employ structure-based testing to thoroughly check all the constructs in the code.  
  
For example, in an inventory control system consider that you have a function with a decision construct that adds new items to different tables based on whether they are fast moving or slow moving. To check the structure and function of the decision construct used, you employ structure-based techniques.

Typically, low-risk or low budget systems may lack clear specifications or a well-defined structure. Creating test cases in such instances can become daunting with the lack of adequate information or specifications.   
  
So the perspective and expertise of testers, developers, and other stakeholders with experience working on similar software to design test cases is useful based on their past experience and previous knowledge. This technique is known as experience-based testing.

Using these techniques is also useful when you need to create test cases in a short time and when testing the structure or completeness is not critical.

Question

As part of a testing team, you need to create test cases for a cash management system to be implemented at a fast food restaurant. The code is already finalized and your client has specified the code be exercised completely at this checkpoint. Identify the dynamic test technique you will use.

**Options:**

1. Specification-based techniques
2. Structure-based techniques
3. Experience-based techniques
4. Static Analysis technique

Answer

***Option 1:****Incorrect. Specification-based techniques cannot be used to check the structure of the code. You use these techniques when you want to create test cases using the specifications.*

***Option 2:****Correct. To check the various constructs and the complete structure of the code, you need to use structure-based techniques.*

***Option 3:****Incorrect. These techniques are used to quickly create test cases using the testers knowledge and experience with similar software.*

***Option 4:****Incorrect. Static Analysis is a static testing technique used to check the technical accuracy and validity of source code specified.*

**Correct answer(s):**

2. Structure-based techniques

**2. Summary**

Dynamic software test design techniques include techniques that help test code execution. These techniques are grouped into three categories – specification-based testing, structure-based testing, and experience-based testing.   
  
Specification based testing techniques – also called black-box testing – test system outputs or what the system does. Structure-based techniques or white-box testing test the underlying code and structure that makes a system behave the way it does. Experience-based techniques, on the other hand, enable you to create test cases based on the experience and knowledge that testers, developers, or other stakeholders have about existing or old systems. Although each technique is efficient in detecting specific types of defects, combining them is a good strategy to uncover more defects.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a03_it_enus/output/html/course_transcript.html#top)

© 2016 Skillsoft Ireland Limited