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

**Course Transcript**

Configuration Management, Risks, and Incidents in Software Testing

**Configuration, Risk, and Incident Management**

| [1. Project Risks in Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#t5) |

| [2. Product Risks in Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#t8) |

| [3. Basic Incident Reports in Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#t11) |

| [4. Logging and Managing Incidents in Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#t14) |

| [5. Configuration Management in Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#t22) |

| [6. Managing Risks and Incidents](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#t17) |

Project Risks in Software Testing

Learning Objective

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

* *recognize project risks and the most appropriate way of dealing with such risks*

**1. Identify and mitigate project risks**

Risk is an integral part of life. Every action has some degree of risk associated with it. Even a simple task such as crossing a street could be risky. Similarly, your projects may also face risks because of uncertainty, making risk management an integral part of a project.

Most projects face potential risks that can adversely affect the cost and profitability of a project. The level of risk is characterized by the likelihood of an unfavorable event and its impact on a project.

You can minimize potential risks and maximize potential opportunities by creating a risk management plan. Risk management helps you identify and handle potential risks. It also helps you monitor your project.

Information technology projects involving software testing typically face two types of risks:

**project risks**

Project risks are unfavorable events that may jeopardize the achievement of one or more stakeholders' project objectives on time. For example, introduction of third-party software, developers exceeding their scheduled estimates, application complexity, change in scope of testing, budgetary constraints, or inadequate training may adversely affect the testing schedule.

**product risks**

Any risk that adversely impacts the quality of a product being delivered constitutes product risk. Risks such as failure to deliver software that meets the specifications or client needs, inadequate test processes, software with incomplete or erroneous data, or inadequate competency of testers can affect the quality of software.

Software testing is an important activity in information technology projects where software development is involved. And so, it is subject to risks similar to other activities in the project.   
  
To deal with project risks that may apply to testing, you can ignore the risk, transfer the risk, mitigate the risk, or create a contingency plan.

You may ignore a risk if the probability of its occurrence and impact is low. For example, if a tester goes on vacation during the project or falls ill, you can ignore the risk to the project if you have additional testers who can take on the pending work. However, you need to create a contingency plan to cover a key project team member such as a Test Lead becoming unavailable.

If a risk is highly probable, you can transfer it to a third party. When you transfer a risk, the responsibility for managing the risk is transferred to another party; it doesn't eliminate the risk.  
  
For example, you can outsource a high-risk activity to a vendor who has the expertise to deal with it. Or you can move the risk to someone with a high risk tolerance, for example, an insurance firm.

Another strategy to deal with project risks is to create a contingency plan. Under this plan, risk events are identified before the project begins. A response plan is created that will only be executed if certain events occur under predefined conditions. Events that trigger the contingency response, such as missing milestones or changes in client requirements, are defined and tracked.

You can also mitigate a risk by taking proactive steps to minimize the negative impact of the risk. For example, to mitigate the risk of an experienced member leaving the team, you can maintain a list of current employees or contractors who can step in to fill the vacancy. Taking early action to reduce the potential risk impact is more effective than trying to rebuild a project after the risk occurs.

When creating a risk management plan, you first identify the potential risks. To do this, you can take proactive steps, such as creating a draft of project risks, and receiving input and feedback about the risks from your team members.  
  
These are the common risk types that occur in a testing project.

**Supplier issues**

A testing project may face risks such as failure of a supplier to provide on-time delivery of critical materials for the testing environment. You may also face issues such as failure to consider efforts during the testing process in the contract or inability to respond to the testing issues when raised.

**Organizational issues**

Organizational issuesassociated with a testing project include shortage of skilled testers, inadequate training, outdated development and testing practices, and failure of testers to communicate and respond to test results.

**Product changes**

When there is a change in product requirements, it can result in additional efforts and delayed schedules. It can also result in extensive modifications in existing test cases or creation of new test cases, besides invalidating test results.

**Test environment issues**

Sometimes risks related to inadequate or unrealistic testing environments, such as limited network connectivity, poor system configuration, or unavailability of required applications, result in a delayed project or misleading data. You can either transfer these risks or mitigate them. Setting up a proper test environment enables you to get accurate results and save time.

**Faulty test items**

Test items or test cases that can't be installed properly in a test environment pose a risk for the project. For example, test cases may not target critical areas of an application or the test plan omits tests for an important functional area. To ensure that all test items are properly tested in the test environment, you can create a contingency plan or perform smoke testing before starting the testing process.

**Technical problems**

Risks that arise because of technical problems in the testing project include issues such as difficulty in defining test requirements, constraints that limit the effectiveness of test cases, quality of the product design, or highly complex systems.

Consider a situation when you use the black-box testing environment for acceptance testing. You realize that the new software doesn't install properly over the older version as the configuration files are incompatible. In this case, the test environment is insufficient and so is a project risk.

To mitigate this risk, you need to ensure that a defined uninstall process is part of your project plan. This enables you to clean up the environment after testing and ensure that old components of a system are not left to conflict with new software builds.

Apart from insufficient test environments, you can encounter organizational problems such as ineffective communication or unreasonable expectations from testers.   
  
Suppose the system you're testing is dependent on timely communication between the server and its clients. To ensure that this happens, you need to measure the response time for client requests. You also need to conduct the test when the clients and the server are on the same infrastructure. However, you realize that the server and the clients are geographically distant. This is a problem since the plan does not contain any provision for testing from geographically distant locations.

To mitigate the risks arising out of the problem, you can outsource the tests to other companies that are located in a specified geographical area and can provide more realistic environments. This will help you transfer the risk ownership to a more competent company.

You can also manage risks related to changes in product during the development stage.   
  
Consider that you are testing a graphic manipulation library for use in a variety of applications. The development team's priority is to deliver a feature-rich product that meets client needs. Although the test plan is complete and the project is moving into the acceptance testing phase, the developers have received new requirements and have added a new text manipulation and rasterization functions to the library. These functions are not part of the designed test cases and thus pose a project risk.

You can then mitigate this issue by implementing product change-control processes, robust test design, and light-weight test documentation, allowing for more flexibility.

In severe incidents, when new features are added repeatedly, it may be necessary to transfer the risk by escalating the issue to management.

Question

You're testing a web application. The test environment requires a robust Internet connection with a minimum speed of 1Mbps to execute the test cases. However, the IT department in your organization can only provide 500Kbps Internet speed. This makes your test environment insufficient for performing test cases.  
  
What are the most appropriate ways to deal with this risk?

**Options:**

1. Request your client to change application requirements so the application can be tested on a lower Internet speed
2. Transfer the risk responsibility by escalating the issue to management
3. Outsource the testing activity to an outside vendor with the required Internet connectivity
4. Test the application on the Internet speed available in your company

Answer

***Option 1:****Incorrect. It is not the recommended mitigation plan for test environment issues. Because the application requirements are already specified and accepted, you can't ask the client to change the requirements in the course of a project.*

***Option 2:****Correct. Because, you may not have enough authority to decide on increasing the Internet connectivity, you can transfer the risk associated with insufficient or unrealistic test environments by escalating the issues to the top management. The management then decides the resolution for such issues.*

***Option 3:****Correct. You can outsource the testing activity to an outside firm with the required test environment. This will save you the huge fixed cost associated with taking an additional Internet connection.*

***Option 4:****Incorrect. This is not an appropriate resolution of the issue because you will not meet the requirements of the application and the application can show errors when tested in future.*

**Correct answer(s):**

2. Transfer the risk responsibility by escalating the issue to management  
3. Outsource the testing activity to an outside vendor with the required Internet connectivity

Question

Suppose a part of the system that you're testing depends on a Commercial-Off-The-Shelf (COTS) web proxy that needs to be installed in the test environment prior to testing. This proxy was ordered several weeks ago, but still hasn't arrived. Although you have contacted the vendor several times, they insist the product will arrive soon.  
  
What is the risk type in this situation?

**Options:**

1. Test item
2. Organizational issues
3. Product changes
4. Supplier

Answer

***Option 1:****Incorrect. Risks related to test items occur when you're unable to properly install a test item in a test environment. It can also occur if a test item doesn't target critical functionality of an application.*

***Option 2:****Incorrect. Organizational issues include shortage of skilled testers, inadequate training, outdated development and testing practices, and failure of testers to communicate and respond to test results.*

***Option 3:****Incorrect. When there is a change in product requirements or features during the testing phase, it can result in product change risks.*

***Option 4:****Correct. This is a supplier risk because there is a delay from the vendor in providing critical resources for the test environment.*

**Correct answer(s):**

4. Supplier

**2. Summary**

Every testing project is prone to certain risks. You can manage the risks in software testing by ignoring them or creating a contingency response plan. You can also transfer the risks to others capable of handling them or try to mitigate the risks when they occur.   
  
Some typical risk types that occur in a testing project are related to supplier, organization, test environment, product changes, faulty test items, or technical problems in the product.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#top)

Product Risks in Software Testing

Learning Objective

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

* *determine the priority of product risks*

**1. Risk-based software testing**

Software applications, like all products, are subject to risk; they are prone to failures and can dissatisfy customers, end users, and stakeholders. Although conventional testing helps prevent failures, it concentrates equally on all features of the application. As an alternative, you can conduct a risk-based test. In this test, features that are more likely to fail and features that have a bigger impact on business are given higher priority during testing. This ensures that the risk of failure for the application is minimized.

If a bank application, for example, calculates lower interest, the bank would incur losses on the loans provided. However, if it displays the incorrect time, it wouldn't affect the business of the bank. So if you conduct a risk-based test, you'd prioritize the testing of the interest calculator over the clock because its failure would prove more expensive.

Similarly, if you're aware that the feature that transfers funds is likely to fail because it was poorly coded, you'd test it before testing the interest calculator.

Consider that you want to implement risk-based testing for a web application of a department store. To do so, you create a risk analysis table. In the table, you first list the features of the application.

Graphic

*You've listed five features of the web application in this example in the first column of the table. These five features are View Items, Shop Online, Login, View 3D Image, and Shopping Cart.*

You then specify the probability of failure for each feature using values such as High, Medium, and Low. A value is assigned to a feature after assessing the complexity of the feature, using existing knowledge of the application, and considering the expertise of the developers. You can also use the data of previous projects that tested a similar application to determine a value for each feature.

Graphic

*You label the second column of the table as Probability of Failure.*

Note

*You can also use other scales such as the five-grade scale that contains the values Very High, High, Medium, Low, and Very Low.*

You specify a probability of failure value for each feature of the department store web application. The features you can specify in the risk analysis table include

**View Items**

The probability that the View Items feature will fail is low because it was created by an experienced development team.

**Shop Online**

The probability that the Shop Online feature will fail is high because it's a complex feature; it needs to communicate with several applications, such as the payment gateway and bank applications, to complete a transaction.

**Login**

The probability that the Login feature will fail is medium because this feature is mildly complicated; it needs to contact the server to verify the login details of customers.

**View 3D Image**

The probability that the View 3D Image feature will fail is low because this feature was created using a template used in a previous project.

**Shopping Cart**

The probability that the Shopping Cart feature will fail is medium because this feature was created by two different development teams.

You then specify the impact of failure; you determine what impact each feature will have on the customer's business if it fails.  
  
In this example, the View Items and Shop Online features are critical and the store would lose business if they fail. So the impact of failure for these features is specified as High.

Graphic

*You label third column as Impact of Failure and specify the impact of failure values in this column.*

If the Login feature fails, the application can still be used. However, if customers can't login, details such as the address and credit card number won't be retrieved automatically. Customers would find it inconvenient to enter these details before purchasing every item and won't shop as much as they want to, so the impact of failure for this feature is specified as Medium.

The impact of failure for the Shopping Cart and View 3D Image features is specified as Low because they're additional features that don't affect the store's business.

Next you calculate the risk priority value for each feature. This value is calculated by adding the probability of failure and impact of failure values. For example, if you assume the values of High, Medium, and Low are 3, 2, and 1 respectively, the risk priority value of the Shop Online feature would be 6. Similarly, the risk priority value of the Shopping Cart feature would be 3.

Graphic

*You label fourth column as Risk Priority and specify the risk priority values in this column. The risk priority value of the View Items feature is 4, Shop Online feature is 6, Login feature is 4, and View 3D Image feature is 2, and Shopping Cart feature is 3.*

Note

*As an alternative to the probability and impact values being added together, it is also common practice to multiply them to get a risk priority value. In the following instruction and exercises, we assume that they are added.*

Finally, you arrange the features in the descending order of their risk priority values to complete creating the table.   
  
When two features have the same risk priority value, you test the feature that has a higher impact of failure value first. In this example, both the View Items and Login features have a risk priority value of 4. However, the View Items feature is placed above the Login feature because its impact of failure value is High.

During the testing process, you test the features in the order they're arranged; you begin by testing the feature on top of the table.

You can also increase the scope of the risk analysis table to include global attributes such as performance and security. You can then specify probability of failure and impact of failure values for these attributes and determine which attribute should be prioritized during testing.

Additionally, you can use risk priority values to decide which features and attributes require rigorous testing and which require minimal testing. Rigorous testing can include activities such as testing the source code and creating a prototype of the feature.  
  
For example, you can decide that features with a risk priority value of 5 and above should be rigorously tested. Similarly, you can decide that features with a risk priority value of 2 should be minimally tested.

Question

You're creating a risk analysis table for an inventory tracking system. Assuming that the values of High, Medium, and Low are 3, 2, and 1, respectively, how would you prioritize the testing of the Remove Item feature?

*You've specified that the probability of failure and impact of failure values of the Add Item feature is Medium and High, Remove Item feature is Medium and Low, Edit Item feature is Medium and Medium, Display Item feature is High and High, and Copy Item to CD feature is Low and Low.*

**Options:**

1. Test it after the Display Item feature
2. Test it after the Display Item, Add Item, and Edit Item features
3. Test it after the Display Item and Add Item features
4. Test it last after the Display Item, Add Item, Edit Item, and Copy Item to CD features

Answer

***Option 1:****Incorrect. You should test the Add Item feature after the Display Item feature.*

***Option 2:****Correct. The risk priority values of the Display Item, Add Item, Edit Item, Remove Item, and Copy Item to CD features are 6, 5, 4, 3, and 2 respectively. So you should test the Remove Item feature after the Display Item, Add Item, and Edit Item features.*

***Option 3:****Incorrect. You should test the Edit Item feature after the Display Item and Add Item features.*

***Option 4:****Incorrect. The risk priority value of the Copy Item to CD feature is the lowest. So you should test this feature last.*

**Correct answer(s):**

2. Test it after the Display Item, Add Item, and Edit Item features

Question

You're creating a risk analysis table for a basic text editor. You've already specified that the probability and impact of failure values of the Add Text feature is Very Low and Very High, Delete Text feature is Very Low and Medium, Save File feature is Very High and Very High, Open File feature is Medium and High, and Spell Check feature is High and Very Low. Arrange the features in the order you'll test them. Assume that the values of Very High, High, Medium, Low, and Very Low are 5, 4, 3, 2, and 1 respectively.

**Options:**

1. Save File
2. Open File
3. Add Text
4. Spell Check
5. Delete Text

Answer

**Correct answer(s):**

**Save File is ranked**

The risk priority value of the Save File feature is 10, so you should test it first.

**Open File is ranked**

The risk priority value of the Open File feature is 7, so this should be the second feature that you test.

**Add Text is ranked**

The risk priority value of the Add Text feature is 6, so this should be the third feature that you test.

**Spell Check is ranked**

The risk priority value of the Spell Check feature is 5, so this should be the fourth feature that you test.

**Delete Text is ranked**

The risk priority value of the Delete Text feature is 4, so you should test it last.

**2. Summary**

In risk-based testing, features that are more likely to fail and features that impact business are tested before other features.  
  
To implement risk-based testing, you create a risk analysis table. In this table, you list all features and specify the probability and impact failure for each feature. You then calculate the risk priority value for each feature. Finally, you arrange the features in the descending order of their risk priority values. During testing, you begin by testing the feature on top of the table.  
  
In addition to features, you can also add global attributes such as performance and security in the risk analysis table. Also, you can decide which features and attributes should be rigorously tested and which should be minimally tested.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#top)

Basic Incident Reports in Software Testing

Learning Objective

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

* *recognize the purpose of an incident report and what it contains*

**1. Incident reports**

Testing software may produce different results from those expected. Such a result is called an *incident*. For example, according to the functional specification document, you should be able to generate reports while testing billing software, but the application hangs when you try to generate a report. This result is different from the expected and so is an incident.

If an incident wouldn't cause any problem to the software, then you can leave it as is. However, if the incident causes the software to malfunction, you can categorize it as a *defect*. In this case, the developers need to rectify the defect so that the software functions as expected. Defects can appear in any phase of the development process of the software.

Incidents occur for various reasons. An incident could arise if the test environment is configured incorrectly or if the tester makes mistakes while testing. Logging incidents helps you take appropriate action at the right time. You log or document the details of an incident or a defect in an *incident report*.

An incident report also enables you to categorize the incident. For example, you can categorize the incident as a bug, an observation, or an enhancement.

You use the incident report to log an incident for various purposes. The report helps you

**provide feedback to developers**

You should provide adequate and complete information in the incident report so that it provides feedback to the developers. This feedback would enable them to locate and correct the defects or to handle the incident in an appropriate manner.

**track the quality and progress of testing**

The incident report provides the managers and test leads a general overview of the product quality. For example, if the number of incidents or defects is numerous, then the quality of the application is poor.  
  
The report also helps them understand the areas that contain the most errors and the severity levels of these errors. Say for instance, when testing an online web site, you report various incidents and defects on the look and feel of the web pages. This means the developer needs to revisit the web site to change some aspects of the graphical design.

**improve the test process**

You can improve the test process by determining the phase in which the incident or defect was detected. For example, if you test an online web site from the design until the deployment phase, you'll find incidents and bugs at all phases. If you analyze them, you'll get consolidated data about the number of bugs in each phase. Then you can perform additional tests in each phase and remove the defects. This helps improve the efficiency of the test process.

A typical incident report is a template that contains four main sections: Test Incident Report Identifier, Summary, Incident Description, and Impact.  
  
You provide a unique ID for the incident in the Test Incident Report Identifier section.

In the Summary section, you provide a brief description of the incident. For example, you can explain the variation observed between the expected and actual result while testing the software. In addition, you can provide high-level detail of the application you tested and the incidents you came across while testing.

You describe the incident in the Incident Description section. It contains subsections such as

**inputs**

It is important to record the input data that triggered the incident. For example, for a tax calculation application, if you enter the salary amount in four decimals digits, the application might round off the tax value to display two decimals. In such cases, you should note that your input value was a four decimal digit.

**expected results**

You should document how the software application is expected to function. For example, you want the tax calculation application to produce results in four decimal digits. Therefore, you specify the same in the expected results subsection of the Incident Description section.

**actual results**

After you execute the application using the test data provided, the actual result you observe should be documented in the incident report. This could either be the same as the expected result or could be different. For example, the actual result would be different if the salary value expected contains four decimal digits and in the output displays two decimal digits.

**anomalies**

Any anomalies – how the actual result differs from the expected result – or an unusual occurrence you find when executing the software application should be documented. For example, you test the tax application by entering the salary in fractions. You find that the software doesn't respond. This is an anomaly and so you should document the values that caused the anomaly.

The other subsections in the Incident Description section include

**the date and time of the incident**

You can log the date and time when the incident occurred in the Date and Time section. However, if you're using a testing tool to log incidents and defects, this information is automatically captured.

**the procedural step**

The developer who fixes the incident or the defect should be able to check the application and make appropriate corrections in the software application. So the specific procedural step that caused the incident should be documented in the incident report.

**the environment**

You have to document the environment in which you tested the application. Environment includes hardware and software used for executing tests. For example, according to the test specifications, you have to test the application on Apple Macintosh platform and so you have to specify this information in the incident report.

**the number of attempts to repeat an incident**

If you find an incident, you should be able to repeat it. You should also document the number of times you attempted to repeat the test. You should be able to document the exact steps that caused the incident so if somebody else tests the application, the same incident should occur again. The developer can then deduce the problem and correct it.

**comments**

As a tester and observer, you can provide general comments. For example, you can specify if a section of the application code is modified based on the incident then all the resultant codes should also be checked for the same change. You can also specify any global issues that should be taken care of while fixing the application in the Incident Description section.

After providing the details in the Incident Description section, you can use the Impact section to describe how the incident would affect the end user. Depending on the severity of the impact, you can categorize an incident as one that would have a low, medium, or high impact. You can also rate the incidents on a scale of one to five or ten, and based on the severity level or rating, you can assign an order of priority to each of them.

Supplement

*Selecting the link title opens the resource in a new browser window.*

**Learning Aid**

Use the learning aid [Test incident report template](javascript:doWindow('./lasd_sftf_a07_it_enus_t1101_frame.html')) as a summary of the various sections in an incident report.

Question

What are the purposes of an incident report?

**Options:**

1. Provides feedback to the developers
2. Helps the senior management decide whether the project was profitable
3. Measures the quality of the product
4. Helps improve the test process
5. Enables you to determine the root cause of the defects

Answer

***Option 1:****Correct. Based on the information in incident report, the developers can correct the defect or handle the incident in an appropriate manner.*

***Option 2:****Incorrect. The objective of an incident report is to help determine the quality of the product or application and doesn't determine whether the project is profitable.*

***Option 3:****Correct. Depending on the number of incidents or defects logged, you can determine the quality of the product.*

***Option 4:****Correct. The objective of creating an incident report is also to improve the test process. This is possible if you can determine the phase in which the defects occurred.*

***Option 5:****Incorrect. After logging an incident, you need to determine whether it is a defect. And only if it is a defect, do you need to determine its root cause.*

**Correct answer(s):**

1. Provides feedback to the developers  
3. Measures the quality of the product  
4. Helps improve the test process

Question

What details does the Incident Description section of an incident report template contain?

**Options:**

1. A brief overview of the incident
2. Details on how the incident affects the project timelines
3. Date and time of the incident
4. Actual results noticed after testing

Answer

***Option 1:****Incorrect. You provide a high-level summary of the incident in the Summary section of the incident report template.*

***Option 2:****Incorrect. You provide a brief description about the impact of the incident on project timelines in the Impact section of the incident report.*

***Option 3:****Correct. It is important to enter the date and time when the incident was logged. This would help you to track any issues at a later point of time.*

***Option 4:****Correct. The actual results observed when testing the application is documented in the Incident Description section of the incident report. This could be different from the expected results.*

**Correct answer(s):**

3. Date and time of the incident  
4. Actual results noticed after testing

**2. Summary**

An unexpected behavior that occurs while testing an application is called an incident. You can further analyze incidents as defects.  
  
You log incident details, such as the phase in which the error occurred, in an incident report. By logging the incident details, you provide feedback to the developers about the application, track the progress of the testing process, and improve the test process.  
  
An incident report contains four sections: Test Incident Report Identifier, Summary, Incident Description, and Impact. You specify the unique ID of the incident report in the Test Incident Report Identifier section. In the Summary section, you provide a brief description of the incident, and in the Impact section, you describe how the incident would affect the progress of testing.   
  
You can specify various details such as inputs provided, results expected, and actual results in the Incident Description section.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#top)

Logging and Managing Incidents in Software Testing

Learning Objectives

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

* *recognize how the incident report process works*
* *recognize what is included in an incident report*

**1. Elements of incident report template**

Logging the outcome of testing a product is as important as testing itself. This ensures that the product evolves with the desired quality through a systematic testing process. A typical test log contains incidents, the unanticipated results that you observe when executing an operation. You document incidents in an incident report.

One type of unanticipated result is a defect that can cause system failure by preventing the system from functioning according to its specifications. For example, online banking software may fail to execute the operation of printing the statement when the user clicks the "Print Statement" option.

Such failures arise due to individual defects, also called bugs. You need to prepare a test incident report, recording not only defects, but also any other incidents that are not as critical as bugs.

You use specific defect-tracking tools to record incidents, and the Institute of Electrical and Electronics Engineers (IEEE) Template to create a test incident report.

To record incidents, you can use the various elements in the IEEE Template. It includes

**incident report identifier**

You assign a unique incident report identifier to your test incident report. This identifier uses a numbering scheme to track incidents and enables you to view the report of each incident.

**incident summary**

You use the incident summary to relate an incident to the test case in which the incident was detected. A developer reads the incident summary and recreates the situation to run the application in an environment similar to that in which the bug was identified. You then run the test case again to ensure the developer has fixed the defects.

**incident description**

The incident description provides information on incidents to the readers and enables them to replicate each incident from a user's perspective. Usually this description comprises details such as inputs, expected results, actual results, testing date and time, procedure followed, environment, the number of times the test case was run, and the tester's comments, if any.

**impact**

You document the potential impact of the incident on the users under the impacts section. This information also helps you to prioritize the bug fixes. To measure the impact of defects, you use a standardized impact rating scale. For example, you use a scale with categories such as Minor, Major, and Critical. The categories must be well-defined so that the impact rating is uniform and it doesn't depend on the tester who assigns the ratings.

From the elements you used in the incident report, the developer recognizes the root cause, the phase of the project in which the incident was identified, and the action that produced the incident. In addition, the incident report provides references to documents specifying the correct behavior.

Question

Match each element of the IEEE incident report template to its most appropriate use.

**Options:**

1. Incident description
2. Incident report identifier
3. Impact
4. Incident summary

**Targets:**

1. Enables you to view the incident report using a unique numbering scheme
2. Replicates incidents provided by the user
3. Tracks each incident back to the test case in which it was detected
4. Helps to assign priority rating for bug fixes

Answer

*The unique incident report identifier you assign to each of your test incident reports helps you to identify and view the report of each incident.*

*Incident description comprises details about inputs, expected results, actual results, testing date and time, procedure followed, and environment. You can replicate the incidents using these details provided from a simulated user end.*

*A developer goes through the details provided in the incident summary and recreates the situation under which each incident was noted.*

*The impact section enables you to prioritize bug fixes based on the impact of each bug on the user.*

**Correct answer(s):**

Target 1 = Option B

Target 2 = Option A

Target 3 = Option D

Target 4 = Option C

Question

Suppose you are creating a test incident report. Which elements should you include in your report?

**Options:**

1. Share input and result details
2. Assign a unique identifier
3. Specify a format to add incident report elements
4. Prioritize bug fixes and rate the impact of each bug
5. Provide details to track incidents back to test cases

Answer

***Option 1:****Correct. You add the incident description to share details such as inputs, expected results, actual results, testing date and time, procedure followed, and environment.*

***Option 2:****Correct. You assign a unique incident report identifier to each of your test incident reports so that you can easily locate and access each incident and its report.*

***Option 3:****Incorrect. The Institute of Electrical and Electronics Engineers (IEEE) template is the specified format on which you add the different elements of your test incident report.*

***Option 4:****Correct. You need to prioritize bug fixes based on the impact of each bug on the user. You follow a predefined impact rating scale to measure the impact of defects.*

***Option 5:****Correct. In the incident summary, you provide details to enable the developer to track an incident back to the test case in which you detected the incident. The developer goes through the incident summary and recreates the situation.*

**Correct answer(s):**

1. Share input and result details  
2. Assign a unique identifier  
4. Prioritize bug fixes and rate the impact of each bug  
5. Provide details to track incidents back to test cases

**2. Working of the incident report process**

Testing of each incident goes through a series of steps before the incident can be verified and closed.  
  
The stages involved in the life cycle of an incident report are shown in this flow chart.

Graphic

*Description of the incident report life cycle:  
  
The flow chart starts with Reported stage of the incident report life cycle. The Reported stage has two branches – if rejected, it reaches the Rejected stage. If accepted, it reaches the Opened stage. If approved, the report reaches the Assigned stage. If the project team declines it for repair, the report is Deferred. The assigned report then reaches the Fixed stage. It then reaches the Closed stage, or if not fixed, reaches the Reopened stage.  
  
Description ends.*

**Reported**

When you log a test incident report, an incident is in the **Reported** stage. A peer tester or test manager reviews the incident report in this stage and decides if it should be opened or sent back for rewriting.

**Rejected**

If the report is not reproducible or if the defect couldn't be justified, the peer tester or test manager marks the report as **Rejected** and returns it to the tester. The test manager demands more information such as the number of times the tester attempted to replicate the intermittent defect or the number of times the defect was noted.

**Opened**

The incident report is in the **Opened** stage when it passes the initial review and the project team is still working on filtering the defects that they actually need to repair.

**Deferred**

An incident is in a **Deferred** stage if the project team considers it as a defect and wants to decline the defect for repair at least temporarily.

**Assigned**

An **Assigned** defect is the one that the project team approves for repair. Then a programmer begins fixing the defect.

**Fixed**

After repairing the defect, the programmer sends the **Fixed** report back to the tester. The tester performs a confirmation testing based on this report.

**Reopened**

In the event of a failure observed in a confirmation test or a problem being observed after an incident is closed, it must be **Reopened** and reassigned.

**Closed**

The incident report is **Closed** when the tester confirms that the defects are fixed.

Suppose you are testing a multimedia player application developed for mobile phones. You notice that there is a problem playing .3gp files and that the volume level is very low when you play .mp3 files. You first create an incident report by recording the incidents, classifying them, and identifying the impact of each defect.

Your test manager validates the report and opens it for the project team to approve and assign the incidents to the development team.

Then the video developer goes through your description of the incident and attempts to identify and eliminate the cause of the failure while playing .3gp files. Similarly, the audio developer works on disposing of the incident responsible for a low volume.

Once the developers have fixed the defects, they mark the incidents as Fixed and send the updated incident report back to you.

You perform a confirmation test to make sure the application doesn't contain the reported incidents. Now suppose you find that while the defect in running .3gp files is solved, the volume level of .mp3 files is still low compared to that in files of other formats.

You reopen the incident in the incident report and send the report for fixing, if possible, with an additional description of the incident, so that the developer understands the incident better. Then the incident report once again goes through the same process of review and assignment.

The developer continues to work on fixing the defect. If improving the volume level of .mp3 files is not possible at this level, the incident is deferred until the end of the project. If the defect could be fixed, the developer updates the incident report and sends you the updated report. You once again perform a confirmation test, and on confirming the update, you close the incident report.

All the various stages in the testing process – from reporting to closing incidents, except Rejected, Deferred, and Closed – have a distinct owner who is responsible for moving the incident to the next stage. To maintain a smooth workflow, you need a defect-tracking system that can be customized to match your operational requirements.

You can use a defect-tracking tool that incorporates all the different stages to manage incidents. Many organizations develop their own defect-tracking tools and customize it to match their requirements. This helps them save time and effort required to detect and fix bugs.

The defect-tracking tool also enables testers to communicate with managers by providing information related to the number, severity, and status of bugs.

To ensure a smooth workflow and effective incident management, the major attributes your defect-tracking tool should include are

**user-friendly interface**

Your defect-tracking tool should have a user-friendly interface, which doesn't tax the user by demanding excessive information. An ideal defect-tracking tool saves time required to generate an incident report.

**customizable fields**

Customizable fields in a defect-tracking tool enable you to fulfil the requirements of your organization. You should be able to customize the fields in the tool and create categories your organization uses, for example Critical, Major, and Minor.

**discrete data analysis**

With discrete data analysis features, a test manager can easily get the required data in the preferred format, for example a table or a chart. Discrete categories such as type, status, and distribution facilitate easy and quick data analysis.

**dynamic incident log**

A dynamic incident log for each incident recorded in the report enables you to track the progress of the incident through the various stages of its life cycle.

**organization-wide accessibility**

The defect-tracking tool should be accessible across the organization so that all the users are able to access it at any time. This reduces the amount of time spent waiting for someone with more access to move the incidents from a completed stage to the next stage.

Question

You are testing an audio editing application. You noticed that there is a problem saving files as Audio Interchange File Format (AIFF). You create an incident report, which was reviewed, approved for repair, and repaired. What happens to the defect after this stage?

**Options:**

1. Confirmed to be repaired
2. Deferred
3. Closed
4. Reopened for fixing

Answer

***Option 1:****Correct. Once you get back the updated incident report, you perform a confirmation test to make sure the application doesn't contain the reported incidents any more.*

***Option 2:****Incorrect. You defer an incident only when the project team declines the defect for repair due to non-availability of any solution to fix the defect.*

***Option 3:****Incorrect. The defect is closed only after a confirmation that it has been fixed. Generally this is the last stage in the life cycle of an incident report.*

***Option 4:****Incorrect. You don't reopen a defect for fixing unless the assigned defect is found to be unrepaired.*

**Correct answer(s):**

1. Confirmed to be repaired

Question

You are a test manager. You review a test incident report and find that some of the defects reported are intermittent and the tester has not mentioned this in the report. What should you do in this situation?

**Options:**

1. Open the report and pass it on to the project team for fixing
2. Reject the report so that the tester can rewrite it
3. Defer the defect until you receive the missing information
4. Assign the defects to appropriate project team members

Answer

***Option 1:****Incorrect. You open forward an incident report to the project team only when there are no intermittent defects in the report.*

***Option 2:****Correct. You reject a report with intermittent defects and ask the tester for additional details such as the number of times the tester attempted to replicate the defect and the number of times the defect was actually noted.*

***Option 3:****Incorrect. A defect is deferred when the project team decides not to repair the defect due to non-availability of any solution to fix the defect.*

***Option 4:****Incorrect. You assign a defect to the project team only after you approve the incident report. When you find that a report contains intermittent defects, you reject the report and send it back to the tester.*

**Correct answer(s):**

2. Reject the report so that the tester can rewrite it

**3. Summary**

Logging and managing incidents plays a vital role in developing a defect-free product. You need to prepare a comprehensive and easy-to-use test incident report that records all the test cases and related events. You follow the Institute of Electrical and Electronics Engineers (IEEE) template to create a test incident report. This template comprises elements such as incident report identifier, incident summary, incident description, and impact.  
  
The test incident report you create goes through a series of stages such as Reported, Opened, Assigned, Fixed, and Closed. This process continues until all the reports are closed or deferred due to valid reasons.  
  
To record and track incidents, you use a defect-tracking tool. The tool you use should be easily accessible by all the members involved in the product development. It should enable you to enter discrete values in its customizable fields.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#top)

Configuration Management in Software Testing

Learning Objective

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

* *recognize how configuration management works in software testing*

**1. Configuration management**

A software product is likely to go through many iterations during the course of its lifecycle due to changed requirements, changes in functionality, and changes as a result of testing. In such cases, it becomes important to identify and record the configuration items such as the operating system, networking elements, and the source code of the software.  
  
You also need to ensure that changes in each item are properly recorded, reported, implemented, and verified throughout the project. This elaborate process is called *Configuration Management*.

A critical component of configuration management is the use of *version control* to manage configuration items as they go through iterations.   
  
Version control requires you to assign a unique name, a version number, and other attributes to each configuration item. By mapping incidents against version-controlled configuration items, you can easily identify and trace incidents back to the test case during the actual testing.

It also avoids confusion during testing. For example, software is updated whenever a new incident is identified and fixed leading to multiple versions. If the item fixed is not version controlled and documentation is not updated, it is very likely that subsequent testing will continue on the older version of the software, and the same incident may be reported again.

Note

*To avoid such confusions, you can follow the Institute of Electrical and Electronics Engineers (IEEE) template IEEE 829.*

Configuration management involves two functions.

**Library Management**

As part of the configuration management process, library management involves managing and documenting the changes made in the product, versioning the software, and maintaining baselines of the software for easy tracking. A configuration manager, who is in charge of this activity, may perform these tasks using commercial or in-house library management tools.

**Change Control Board (CCB)**

The Change Control Board (CCB) analyzes the incidents reported during testing and decides how to handle them – whether to treat them as a critical or minor change, or as an enhancement that can be carried out in later releases. Typically, this board comprises of end users, testers, developers, and other stakeholders of the product, so that decisions on how to handle each incident are discussed and agreed upon from every perspective.

In addition to rating the impact and prioritizing incidents, the CCB is responsible for conflict resolution and release management. The CCB members meet at regular intervals to analyze incidents and enhancement requests. During analysis of the incident report, they may categorize the incident as a defect or as a change. Regardless of whether defects are being fixed or changes to requirements are being authorized, it is essential that configuration management practices are in place to ensure traceability.

Question

Which of these comes under configuration management in software testing?

**Options:**

1. Implementing version control
2. Creating test incident report
3. Fixing the reported incidents
4. Operating Systems

Answer

***Option 1:****Correct. Version control is a critical component of configuration management. You need to assign a unique name, a version number, and other attributes to each configuration item. This enables you to effectively manage configuration items as they go through iterations.*

***Option 2:****Incorrect. Creating a test incident report is not a part of configuration management. Usually a tester creates the report by listing the observations made while testing the product. Configuration management deals with analyzing the report and assigning the incidents to appropriate resources.*

***Option 3:****Incorrect. The task of fixing the reported incidents is assigned to the developer. The role of configuration management is to track the changes and status of incidents across the various stages of their life cycle.*

***Option 4:****Correct. The configuration of the operating systems used throughout software development and testing should be managed and recorded.*

**Correct answer(s):**

1. Implementing version control  
4. Operating Systems

Question

Which tasks are performed by the Change Control Board (CCB)?

**Options:**

1. Resolving technical conflicts
2. Versioning the software application
3. Determining the severity of incidents
4. Tracking and managing product changes
5. Prioritizing incidents to be fixed

Answer

***Option 1:****Correct. The CCB is responsible for resolving technical conflicts that may arise during the product development.*

***Option 2:****Incorrect. Versioning the software application is an administrative function that is taken care of in library management.*

***Option 3:****Correct. The members of the Change Control Board determine the degree of impact a defect may have on the development of the product.*

***Option 4:****Incorrect. Tracking and managing changes is an administrative function that needs to be performed in library management.*

***Option 5:****Correct. The Change Control Board determines the level of importance of each reported incident and set priorities based on an accepted rating scale.*

**Correct answer(s):**

1. Resolving technical conflicts   
3. Determining the severity of incidents  
5. Prioritizing incidents to be fixed

**2. Summary**

Configuration management requires you to record, report, implement, and verify changes in each configuration item across project phases.  
  
During testing you identify incidents and manage changes by mapping them against the correct version of the configuration item.  
  
Configuration management includes the library management and Change Control Board (CCB) functions. Library management is an administrative function where changes and product versioning are managed. CCB, on the other hand, determines the severity and priority of each incident.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#top)

Managing Risks and Incidents

Learning Objectives

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

* *analyze and categorize risks*
* *create an incident report using the IEEE Test Incident Report Template*

**1. Exercise Overview**

**2. Analyzing and categorizing risks**

**3. Creating an incident report**

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a07_it_enus/output/html/course_transcript.html#top)

© 2016 Skillsoft Ireland Limited