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

**Course Transcript**

Tool Support in Software Testing

**Tool Support for Testing**

| [1. Benefits and Risks of Tools in Software Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#t2) |

| [2. Tools for Management, Specification, and Static Testing](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#t5) |

| [3. Tools for Test Execution, Logging, and Monitoring](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#t8) |

| [4. Introducing a Tool into an Organization](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#t11) |

| [5. Software Test Tool Classification and Implementation](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#t14) |

Benefits and Risks of Tools in Software Testing

Learning Objectives

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

* *recognize the risks and benefits of automatic test tools*
* *recognize the different roles of data-driven scripts and keyword-driven scripts*

**1. Risks and benefits of test tools**

Software testing involves many repetitive tasks and testers may lose interest in them and make mistakes. Examples of repetitive tasks include testing performed on regressed software, and presenting the test results. When testing regressed software, you need to consider the effort involved in entering the same data or running the same tests repeatedly. After the testing is completed, you have to consider the effort put into the presentation of the test results.

You can automate such tasks using test tools instead. Test tools are software that support test activities such as planning and controlling tests, creating test specifications, building initial test files and data, executing tests, and analyzing test results. You can use a test execution tool to enter the data or run the test repeatedly in the case of the regressed software. And you can use a test management tool to quickly present the results of a test as a chart.

Automating the testing process using test tools helps improve accuracy and reduce costs. It also enables testers to concentrate on other important tasks such as planning, analysis, and design.

Using tools to automate processes, however, is not an easy task. Developing a tool can be expensive. It could also prove to be a difficult task. There are risks associated in purchasing a tool too. You should analyze the tool before buying it, or you risk buying a tool that may not match your requirements.

Even if you do obtain a tool that meets all your specifications, you should also keep the costs associated with using the tool in mind. These costs, such as the maintenance cost, should be less than the cost of testing without the tool. To avoid the risk of incurring loss over your investment and to determine when you can make profitable use of the tool, you can analyze the payback model for test tool implementation.

The graph shows the initial investment of deploying a testing tool as being high, but having a moderate accumulation of maintenance costs over time thereafter. Manual testing on the other hand shows little or no initial investment costs, but over time the ongoing costs accumulate more steeply. The point at which the accumulated costs for both testing methods meet is known as the break-even point; beyond this point the tool has saved on overall costs and the investment proves profitable.

Suppose you spend $75,000 a year on manual testing. You decide to automate the process with a test tool instead. The tool costs $200,000 to implement plus $25,000 a year to maintain and run.  
  
In this example, the break-even point is achieved after four years. So you must use the tool for at least four years before you break even and begin to achieve financial benefits.

Tools provide benefits both in the short and long term. A good software testing tool can potentially

**reduce time and effort for repetitive work**

A test tool reduces the time and effort spent on repetitive work by automating it. For example, you can use a static analysis tool to verify the code against coding standards. Now, verifying the working of the code and checking that the coding standards are followed takes time. Using a static analysis tool instead would help to thoroughly check the code and verify the standards used. Automating these activities will also reduce the time and effort for you.

**provide more predictable and consistent results**

Tools provide consistent and predictable results. Humans, on the other hand, are prone to making errors due to issues such as forgetfulness, misunderstandings, incorrect assumptions, and confusion. For instance, if you're entering test data into a database, you may, unlike a test tool, enter the same data twice or miss entering some data.

**access and present accurate test management information**

Tools can also be used to access and present accurate test management information. Tools can easily retrieve test data from a database and present it as graphs and charts. This makes it easier for your team to understand the data.

**ensure reports or findings are assessed objectively**

When testers calculate values or make estimates from reports, they may not ensure the reports or findings are assessed objectively. They may interpret the data incorrectly, depending on their prejudices. Tools, on the other hand, assess data without any subjective bias. For instance, a tester may become biased when evaluating incident statistics. But, a test management tool makes the assessment consistently without any subjective bias.

Question

Your work involves keying in data or transferring data from one database to another. After each data entry, you verify if data is captured or transferred accurately. For the verification, you have to execute a number of given scripts in the same sequence. While you have to present the results to the manager, you do not have to assess the findings. Your manager wants to buy a test tool and has asked for your suggestions. Predict what benefits you will receive from using a tool in this scenario.

**Options:**

1. Repetitive work will be reduced
2. Costs to the company will be reduced for the short term
3. Ensure reports or findings are assessed objectively
4. Test results will become more consistent

Answer

***Option 1:****Correct. Work of a repetitive nature will be reduced by using a test execution tool. Using the tool, you will be able to quickly verify if the data is captured accurately and is in the correct sequence.*

***Option 2:****Incorrect. A tool will reduce costs only in the medium to long term after it breaks even. In the initial period, the cost of the tool would only cancel the profits.*

***Option 3:****Incorrect. The reports will be assessed by your manager. Your work currently only involves transferring data and verifying the accuracy of the transfer.*

***Option 4:****Correct. Using a tool, your test results would become more consistent because the tool will repeat the scripts in the same sequence each time.*

**Correct answer(s):**

1. Repetitive work will be reduced  
4. Test results will become more consistent

Tools provide many benefits, but there are also risks involved in using them. Some examples of risks include

**underestimating the time, cost, and effort when first introducing the tool**

When you introduce a new tool into an organization, you may incorrectly estimate the time, cost, or efforts required in using the tool. For instance, you give your client a date for delivering a software application. Later, because of technical problems in deploying the tool, and resistance from experienced manual testers, you deliver the software to the client behind schedule.

**expecting more from the tool**

It's possible that team members might expect a tool to do more than it can. For instance, if an organization introduces a static analysis tool to identify potential problems in code, testers may expect the tool to identify problems and also provide suggestions to fix problems. To counter their expectations, you should clearly communicate the tasks the tool can perform.

**underestimating the time and effort needed to derive benefits from the tool**

When you first start using a tool, you may not get the expected results. This is because it takes time and effort to develop ways of achieving the desired benefits. For instance, a tool may have many useful features, but may be complicated to use. Understanding all the features of the tool and availing its benefits may take some time.

Some more examples of risks include

**over-reliance on the tool**

While tools can perform the testing activities they have been created for consistently well, over-reliance on tools can create problems. For instance, tools cannot perform analytical activities. They cannot suggest improvements or evaluate future uses of the tool. The success of a tool also depends upon the skills of the tool user.  
  
When a new product testing requirement comes up, your testing team wishes to use a tool to automate all stages of the testing process. They wish to even automate the tests that take a long time to automate and are best done manually. This is due to the fact that the team is new to the use of the tool and is not skilled enough to make the best use of it.

**underestimating the effort required to maintain the test assets generated by the tool**

A number of tools end up on the shelf after some time because the effort required to maintain the assets generated by the tool was not estimated correctly. The main cause for the incorrect estimation is insufficient planning for maintenance of the assets that the tool produces.   
  
Suppose you use a test execution tool and capture lengthy manual tests. After a while, the assets or the captured tests, would increase and the original space or resources allocated for the tests may not be sufficient. To end the increasing expenditure on the tool, the tool may be put away unused.

Question

You reviewed a proposal for a project and found that the time allotted for the testing process was reduced by 70% because a new test execution tool was to be used instead. Your testing team, which has prior experience in using test tools, is of the opinion that meeting the target would be an easy task if a couple of modifications were made in the existing established testing process. However, you are skeptical about doing so and meeting the deadline. State the possible risks you anticipate in using this tool in the given scenario.

**Options:**

1. Missing the project deadline
2. Releasing a product of inferior quality
3. Testers may have trouble adapting to working with a tool

Answer

***Option 1:****Correct. You run the risk of not meeting your deadline. Because you are using the tool for the first time, your attempts at using it may not be perfect and there may be unanticipated problems.*

***Option 2:****Correct. The quality of the product may suffer because of changes made to accommodate the target. A tool should be brought in to support or improve the existing test process, and not to remove them.*

***Option 3:****Incorrect. The testing team in the given scenario has prior experience in using test tools. So it is not likely that they will have particular difficulties adapting back to using a tool.*

**Correct answer(s):**

1. Missing the project deadline  
2. Releasing a product of inferior quality

The benefits that can be derived or the risks that may be faced when using tools for testing support are also based on the skills of the tester and the skills of the tool user.

The tester should be skilled in deciding what should be tested and how. He or she should be able to read the specifications and decide on the types of testing to be performed and at what levels. He should be able to decide on test cases and ways to create them. He or she should also be able to prioritize the testing activities. On the other hand, the tool user should know which tool to use and how and should also be able to obtain maximum benefits from regular use of the tool.

**2. Roles of scripts**

Some of these tools which use scripting languages require special considerations such as experts to help design tests and script conformance with new coding standards.   
  
These test tools are

**test execution tools**

Test execution tools replay scripts designed to select and execute tests stored electronically. The scripts must be unambiguous for the test execution to understand and execute it.

**performance testing tools**

Performance testing tools test non-functional quality characteristics such as the time taken for a transaction to be completed or the number of users who can log on to and use the system. These tools require an expert in performance testing to help design tests. This expert should know what should be measured, ways of interpreting and presenting the information collected, and the duration of the test.

**static analysis tools**

Static analysis tools are used to enforce coding standards and identify code errors. These tools help create code that is easy to maintain. When these tools are used, there may be issues such as the need to change the old code so that it implements new standards. However, doing so may have some unexpected negative results, which may not be discovered even during regression testing.

**test management tools**

Test management tools support the testing process and make it more efficient. They use other testing tools or spreadsheets to communicate the information or reports they produce in the most effective way. The reports produced need to be monitored to ensure that the reports are most relevant and useful at the time of creation.

Test execution tools may use a variety of scripting techniques. Whichever scripting technique is used, the expected results for each test should be stored for comparison later.  
  
The scripts used by testing tools include

* data-driven scripts, where a control script reads the test data stored in a file or a spreadsheet
* keyword-driven scripts, where information about the test and the control scripts that implement the tests is stored and described in a file
* linear scripts, which are captured by recording the actions of a manual tester or created manually
* shared scripts, which can be reused as a script that can be called by other scripts
* structured scripts, which use selection and iteration programming structures

Data-driven scripts and keyword-driven scripts are the most advanced of all the scripts. In a data-driven approach, all test inputs and expected results are stored in a table or a spreadsheet. A single control generic script then reads and executes all tests in the table and performs these tests using different data. This type of testing is often used to support the application of test execution tools such as capture/playback tools.

In a keyword-driven approach, the spreadsheet contains the test data and the expected results as well as the keywords related to the application being tested. These keywords describe the actions to be taken on the data.

When using the data-driven technique, testers who are not familiar with the scripting language can enter test data only for a set of predefined scripts. These would be hard-coded commands and only the actual inputs would be open for editing.

With the keyword-driven technique, the testers have a number of keywords available that are recognized by the scripts and customized to the application being tested. For example, they may have keyword phrases such as type user, click login, and check name that they can use to define and build their own tests. The keyword-driven technique means that script developers are only called upon when new keywords are needed, and relatively few developers can support a much larger number of testers.  
  
Keyword-driven tools are generally more flexible than data-driven tools.

Question

Which of these objectives are met by either the data-driven or keyword-driven testing techniques?

**Options:**

1. To allow testers to include new commands
2. To allow testers to call the main script from other scripts
3. To allow testers to capture the actions of a manual testing session
4. To allow testers to edit the actual inputs

Answer

***Option 1:****Correct. Keyword-driven scripting allows testers to include new commands in the application. They can do so without rewriting the original script.*

***Option 2:****Incorrect. Shared scripting allows the tester to call a script from other scripts.*

***Option 3:****Incorrect. Linear scripting allows the tester to capture the actions of a manual testing session.*

***Option 4:****Correct. Data-driven scripting allows the tester to edit the actual inputs in a file or spreadsheet so that it can be read by a control script.*

**Correct answer(s):**

1. To allow testers to include new commands  
4. To allow testers to edit the actual inputs

**3. Summary**

Software testing tools support testing processes by automating the test activities. These tools help testers automate repetitive work. They also help testers to plan and execute tests, provide accurate test results, and analyze and present results.  
  
Some benefits of the testing tools are reduced time and effort for repetitive work, more accurate test results, and objective assessment of test findings. Some risks involved in using test tools are unrealistic expectations of users of the tools, underestimation of time, cost or efforts involved when introducing the tools, and over-reliance on the tool.  
  
Some tools such as test execution tools, performance testing tools, static analysis tools, and test management tools use scripting languages to execute their scripts. Test execution tools are capable of using any scripting technique such as linear, structured, shared, data-driven or keyword-driven scripts. Data-driven scripts and keyword-driven scripts are the most important. In data-driven scripting, all the test inputs and expected results are stored in a table or a spreadsheet. In keyword-driven scripting, the spreadsheet also contains keywords related to the application being tested in addition to test inputs and the expected results.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#top)

Tools for Management, Specification, and Static Testing

Learning Objective

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

* *recognize the appropriate use of tools for management, specification, and static testing*

**1. Using test tools for management**

Tools used in software testing can be classified according to the activities they support. Tools designed for the management of the testing process, and for managing tests themselves, come under the Management classification. These include Test, Requirements, Incident, and Configuration management tools.  
  
Management tools help make the testing process more efficient by facilitating the scheduling and management of tests and all testing activities, tracking systems requirements, and logging information about configurations and incidents. For example, requirement management tools provide support to analysts to record and handle the requirements of a system when it is being tested.

The tools for the management of tests and testing procedures include

* test management tools
* requirements management tools
* incident management tools
* configuration management tools

Test management tools manage the testing process. These tools are used by expert testers or test managers during system or acceptance testing.  
  
They contain features that provide support for

**managing testing activities and tasks**

Test management tools provide support for managing testing activities and tasks throughout the development life cycle. They do this by interfacing with other tools or fully integrated modules that provide some or all of the services and functions provided by specialist tools.

**managing test procedures**

Test management tools keep track of test procedures. The test procedures can be stored, edited, linked to requirements, risks or conditions and or reused for future test projects. The procedures are grouped and scheduled to be executed in the most effective manner.

**providing management progress reports based on metrics**

Test management tools can capture and present information on metrics such as incidents raised and the number of tests run, passed, and failed. They can also provide information on the number of defects fixed.

**interfacing with other tools**

Test management tools can interface with test execution, incident management, and requirements management tools. They can also interface with configuration management tools, or alternatively provide their own version control facilities.

Requirements management tools can be used to store, manage, and ensure the consistency and integrity of requirements during the testing life cycle.  
  
You use requirements management tools to

**capture and store requirements**

Requirements management tools capture and store requirements related to test cases and check them for consistency.

**identify defects in requirements**

The tools identify defects in the requirements such as undefined or missing requirements. For instance, some requirements may have issues or hold ambiguous words such as "to be decided" or "and/or". Some requirement management tools will be able to capture these defects.

**identify any changes to other items**

These tools use their traceability function to identify if any changes to other areas of the software that are not under test have occurred as a result of the new requirements. This is made possible through the links and references made between requirements, functions, test conditions, and other testware items.

**calculate requirements coverage metrics easily**

The traceability function of requirements management tools enables test cases to be mapped to requirements quickly, and so requirements coverage metrics can be calculated easily.

Incident management tools manage defects and incidents such as anomalies, enhancement requests, and suggestions that are recorded during testing. These tools are also used to create incident reports.

Incident reports contain details of all stages that incidents pass through such as incident analysis, classification, fixing, retesting, and closing. These reports are stored in a database. The database has field categories such as incidents observed, priority level, severity level, people involved, and current status. The incidents are categorized according to the values stored in relevant fields of the database. As the incidents progress from one stage to another, the values change and the older values are visible in the history of changes made to the incident.

Incidents can also be searched, analyzed, and presented as management information. This information is used along with the data generated by test management tools for planning and estimating new projects and for making process improvements. The information stored can also be used to produce management information about high-priority incidents that are assigned to developers.

Configuration management tools are used to keep track of versions of the software being tested along with the corresponding operating systems and other relevant components, and sometimes the testware itself. They are particularly useful when the architecture of the system you are working on is complex with multiple elements being subject to version change during the testing process.

Suppose your company sells different versions of products that operate on multiple operating systems. You can manage these versions using a configuration management tool that helps you map the version number of each subsystem to the build number of the complete system. You can also map testware depending upon the version number of the build. Mapping allows traceability between the tests and the builds.

Configuration management tools enable you to

**store information**

You can store information about various versions and builds of software and testware.

**trace testware to versions and versions to testware**

By embedding traceability into the system, you can identify the exact configuration of software and testware at any given point of time. You can then determine the correct versions of software, testware and test procedures that need to be reused or corrected when fixing defects.

**perform other miscellaneous activities**

With configuration management tools, you can also manage builds and releases, perform baselining, and control access to information.

Question

Match the functions to the different management tools.

**Options:**

1. Keep a record of the test procedures to be stored for reuse in future projects
2. Identify changes to untested items
3. Store details of software defects
4. Trace testware to versions and versions to testware

**Targets:**

1. Incident management tools
2. Configuration management tools
3. Requirements management tools
4. Test management tools

Answer

*Incident management tools store information on the stages that defects pass through such as analysis, classification, fixing, and closing.*

*When there are any omission issues or ambiguous words such as "to be decided" or "and/or" in the requirements, configuration management tools display warning alerts.*

*Requirements management tools identify if any changes are made to untested areas of the software as a result of new requirements. Through the use of the traceability function, requirements, test conditions, testware, and functions can be linked and referenced. Any changes are then easily identified.*

*Test management tools keep track of test procedures, which are later reused for future test projects. These procedures are stored, edited, or linked to requirements, risks, or conditions.*

**Correct answer(s):**

Target 1 = Option C

Target 2 = Option D

Target 3 = Option B

Target 4 = Option A

**2. Using tools for testing specifications**

The tools available for static testing are

* review process support tools
* static analysis tools
* modeling tools

Review process support tools help keep track of all the information required for a review. These tools are useful for conducting formal reviews where teams located in different locations are involved.

Review process support tools are used to check if there are rules and checklists, if the review comments have been recorded and communicated, if the changes made have been logged, if the changes affect any other items, and if the affected items have been highlighted. They are also used to identify if the review was completed in the planned duration.

Additionally, you can use review process support tools to monitor the status of reviews and store, sort, and communicate reviewer comments. They serve as a repository for all rules, checklists, entry and exit criteria, and procedures used in reviews. These tools also collect metrics on key factors and provide statistical information and reports about comments. Review process tools also provide traceability between comments, documents reviewed, and other relevant documents.

Static analysis tools are usually language-specific tools that come either as part of programming languages or as tools that only work with certain development platforms. With the help of these tools, developers can identify and rectify defects as soon as they occur because these tools analyze code before it is executed and generate warning messages. These tools also identify defects that are hard to find during dynamic testing. Developers use these tools as part of the development and component testing process.

Static analysis tools also help in finding other issues such as syntax errors, invalid code structures, portability issues, security vulnerabilities, references to variables that have null values or stay unused, and inconsistent interfaces between components.

Additionally, static analysis tools are used to identify areas where testing may be required due to additional risks. These tools are also used during activities such as analyzing structures and dependencies and performing static analysis of requirements and enforcing coding standards.

Modeling tools are used by developers during the analysis and design stages of the product development life cycle. These tools help validate models of a system or software. They are used before dynamic tests are run and find omissions, inconsistencies, and defects early in the development life cycle. This helps make development more cost effective and helps developers ensure that they are beginning detailed design from a correct, consistent, and robust model.

Note

*Modeling tools are different from model-based testing tools, which are tools that generate test inputs or test cases from stored information about a particular model. Model-based testing tools are therefore classified under test design tools.*

Modeling tools help

* identify and prioritize specific areas of the model for testing and find inconsistencies and defects in it
* predict system response and behavior of a model under various situations, for instance, under a certain level of stress
* understand system functions and identify test conditions using modeling languages such as Unified Modeling Language

Suppose you are testing a database design. You might discover conflicts arising from columns being present in transaction tables that should be also present in a master table, but are missing. Modeling tools can help you find such inconsistencies or defects early in design that are otherwise costly to pick up later in the development life cycle.

Modeling tools can also check state models or object models. They are particularly useful in complex system architectures. Examples of modeling tools include visual modeling, database, state, and object modeling tools.

The tools available for test specification are

* test design tools
* test data preparation tools

Test design tools help in generating test inputs. When an automated oracle or test basis is available, test design tools also help generate test cases with the expected results. This is why many design tools are integrated with other tools such as test management tools, requirements management tools, static tools, and modeling tools. The details of a test basis are already available in these other tools.  
  
There are different types of test design tools with varied levels of automation. The levels are dependent upon the characteristics of the design tool and the way the test basis is recorded in the tool.

Some types of test design tools are

* tools where test cases with inputs and expected results are generated using specifications in a formal language
* tools that generate tests based on a GUI model of the test basis
* tools that generate a partially completed template from the requirement specifications held in narrative form
* tests that can verify that the model has been built correctly and derive some test cases

Data preparation tools help you collect data, such as fictitious names and addresses, for creating test cases. They are especially useful when you need to collect an extensive volume of data for testing.

Test data preparation tools can generate the data used in tests. They can also select and transfer live data from an existing database and modify it for use in tests. The data acceptable for use can be in a wide range of database formats and files.   
  
The test data preparation tools are used by developers as well as testers during performance and reliability testing. They may also be used for system or acceptance testing.

Test data preparation tools help in

* generating new records with random or guidelines-related data
* sorting or rearranging existing records differently
* extracting selected data records from files or databases
* using a template to construct a large number of similar records for volume tests
* modifying data records for data protection so they cannot be connected to real people

Question

Match the correct list of features with the different management tools.

**Options:**

1. Validates models of a system or software
2. Checks for rules and checklists and communication of feedback
3. Analyzes code even before it is executed and generates warning messages

**Targets:**

1. Review process support tool
2. Static analysis tool
3. Modeling tool

Answer

*A review process support tool checks for rules and checklists, and the recording and communicating of if the review comments. It also checks if the changes made have been logged, and whether they affect anything else.*

*In addition to analyzing code before execution and generating warning messages, static analysis tools also help in finding other issues such as syntax errors, invalid code structures, portability issues, security vulnerabilities, references to variables that have null values or stay unused, and inconsistent interfaces between components.*

*Modeling tools help validate models of a system or software. They are used before dynamic tests are run and find omissions, inconsistencies, and defects early in the development life cycle.*

**Correct answer(s):**

Target 1 = Option B

Target 2 = Option C

Target 3 = Option A

**3. Summary**

Test tools can be classified in several ways. Based on the testing activities they support, test tools can be classified into tool support for management of testing and tests, tool support for static testing, and tool support for management of test specifications.  
  
Tools available for management of tests and testing category include test management tools, requirements management tools, incident management tools, configuration management tools, and test execution tools.  
  
Tools available for static testing include review process support tools, static analysis tools, and modeling tools.  
  
Tools available for management of test specifications include test design tools and test preparation tools.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#top)

Tools for Test Execution, Logging, and Monitoring

Learning Objectives

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

* *recognize the appropriate use of performance and monitoring tools, and test execution and logging tools*
* *recognize the purpose of test execution and logging tools*
* *recognize the purpose of performance and monitoring tools*

**1. Using test execution and logging tools**

As well as providing support for managing tests, test specifications, and static testing, there are other activities such as test execution and logging, and performance and monitoring, that can be supported with the use of tools. To help you perform these activities more efficiently, you have tools that help you execute tests, compare them, and keep them secure. You also have tools to help you monitor a software or system being used, and evaluate its performance in real time.

Some types of tools that enable you to execute and log test cases are

* test execution tools
* test harness/unit test framework tools
* test comparators
* coverage measurement tools
* security tools

Some test execution tools, which are also called capture/playback tools, help you to automatically run test scripts and record their results in a test log.  
  
A test script is a file that is used to test software and compare the actual results to the expected results.

Test execution tools enable you to

* store expected results for comparison when the test is run again
* execute tests from stored scripts and data files accessed by scripts
* compare test elements dynamically and post execution
* record test timings and test results
* send results to test management tools

A test script is essentially program code, and programming skills are required to write or modify scripts. However, many test execution tools facilitate script generation by capturing and recording a tester's or user's manual actions as lines of script. These scripts can then be automated to run repeatedly on the same software module, or modified by a script programmer to utilize different data inputs or to run on different software modules.

However, it is recommended that you don't reuse a captured test script as a long term testing solution because it is likely to be valid only within the exact condition in which it was captured.

Suppose, you record a test script and find that the data values of the original test are embedded in the script itself. This will prevent you from using the original script for another component effectively. While this script can be modified to make it work in most cases, doing so will cause unanticipated delays.  
  
Or, suppose you play back a recorded test script. You then run the risk of the time from the original test being displayed in the current playback. This will result in confusing a test comparator and causing a comparison test to fail.

Test execution tools are especially useful when you need to repeat a test performed earlier for a similar application. For example, you may be able to use a script that captures standard personal details in a number of different software applications.

Though test execution tools are usually used in many forms of testing, they are best suited for regression testing. Here, old tests are rerun to ensure that changes made to an existing system have not altered the functioning of the system.

When used in the medium and long term, the costs of using test execution tools for regression testing decrease. After a point, the costs of automated regression testing become lower than the costs of manual regression testing.

In the graph, the point where the lines converge and cross is the point when the costs of regression testing using test execution tools equal the costs of manual regression testing. In the long run, test execution tools enable you to perform regression tests with better speed and accuracy. They also enable you to avoid manual errors both during execution and during comparison. You can use the time saved to work on the other test-related activities.

A test harness is a test environment that simulates aspects of the production environment in which a software component will eventually run. The test harness contains small programs such as stubs and drivers that interact with the software being tested. Stubs and drivers are examples of unit test framework tools. They are typically skeletal programs designed specifically to call, provide input to, or accept output from, the software modules being tested.  
  
Test harnesses can be programmed in-house using 'XUnit' tools. For instance, you can create a .NET based test harness using an 'NUnit' tool. You can also obtain free source or commercial versions of test harnesses.

A test harness is best used during component or integration testing when parts of the system may be incomplete. Developers usually use test harnesses to test, identify, and localize defects during development.

Test harnesses and unit test framework tools enable you to

* provide input and receive output from the software being tested
* store and execute tests within the framework and record the results of each test
* provide support for debugging
* measure coverage at code level

Test comparators compare what the software produces to what the software is expected to produce and store the results either as part of the test case or as computed through a test oracle. For instance, suppose you are executing a test expecting a particular error to be returned during the test. The test execution tool returns an error, but it is not the one you expected. Then to catch the difference and report it, you will use a test comparator.   
  
Test comparators are usually included with the test execution tools and are particularly helpful for regression testing cases.

Your organization is transferring information from an existing database to a new database. You need to check if the data from tables in the old database was transferred into tables of the new database accurately. If done manually, performing this comparison check would take a very long time. But, using the test comparator, you would be able to automate the comparison in a fraction of the time a manual check would take. You could also run the test comparator dynamically during the transfer to issue an alert if any new data appears to overwrite any existing data instead of being copied into an empty field.

The process of identifying differences between the actual results and the expected results for a component under test is called test comparison. Test comparison can be performed in two ways.  
  
They are

**dynamic comparison**

In dynamic comparison, you use a test execution tool to perform the comparison in real time, while the test is executing. This type of comparison helps identify if the compared and the actual results do not match each other during a test. If the results do not match, you can correct the defect or execute other tests. You can also monitor error messages through this method.

**post-execution comparison**

In post-execution comparison, you use stand-alone and internally developed tools to perform the comparison after a test has finished executing.  
  
This type of comparison is useful when comparing a large volume of data.  
  
Many operating systems have file comparison tools for post-execution comparison.

Coverage measurement tools measure the percentage of quantifiable structural elements or 'coverage items' that are covered by a given test suite. Some coverage tools can go further to identify elements that have not been exercised during the test, and even to suggest test inputs that will exercise these elements.

Depending on the programming language and the test techniques in use, coverage measurement can be intrusive or non-intrusive.

Note

*Software cannot be considered 100% tested if it achieves 100% statement coverage. This only means that those elements that the coverage measurement tool succeeded in identifying have been exercised during the test.*

To test an application using a coverage measurement tool, you

**instrument the code**

You first instrument the code by identifying the coverage items at the component test level.  
  
You use the tools to first identify the coverage items that have undergone a structural test. For example, at the component testing level, the coverage items can be code statements or decisions. Or, at component integration level, they can be calls made to functions or modules.

**test the instrumented code**

You then run the instrumented code through a series of manual or automated tests.

**identify the coverage items that were exercised**

Next you use the coverage tool to count the number of coverage items that have been executed by the test suite. You also use it to identify and report the percentage of the exercised coverage items. If the coverage tool you are using is a sophisticated one, you can even use it to identify the test inputs that can exercise paths that include unexercised items.

**remove the instrumented code**

Finally, just before the code goes into production, you remove the instrumented code.

Security testing tools are used to test a system's resistance to security threats such as computer viruses, worms, or denial of service attacks.  
  
These tools support the execution of test procedures to confirm that there are no flaws in the security systems. They test to check if security systems are secure enough to resist external attacks on the network, support software, databases, and source code. They are particularly important if the application being tested is to be used in an unsecured environment, such as e-commerce or e-business and corporate applications on the Internet.

Because the skills required to use security tools are very specialized, you should ensure that only specialists operate them.

Security testing tools can be used to

* identify virus attacks
* identify weak passwords
* check if the system has weak points such as open ports or points of attack
* check if the operations are secure
* check if the system is secured against denial of service attacks and the results of test attacks
* simulate different types of external attacks

Question

Match the definitions with their different test execution and logging tools.

**Options:**

1. Test execution tools
2. Test harnesses
3. Coverage measurement tools
4. Test comparators
5. Security tools

**Targets:**

1. Capture manual tests
2. Compare expected results with the actual results
3. Calculate the percentage of the code structure and the modules and function calls covered
4. Test system against intrusions such as denial of service attacks
5. A test execution environment comprising stubs and drivers

Answer

*Test execution tools capture manual tests and store them. They use stored test inputs and expected outcomes to execute tests automatically.*

*Test comparators compare what the software produces to what the software is expected to produce. They store the results either as part of the test case or as computed through a test oracle.*

*Coverage measurement tools measure the percentage of the code structure covered and the percentage of the modules and function calls covered. They are used to report coverage measurement and assess exit or test completion criteria.*

*Security tools check if the system is guarded from security intrusions such as denial of service attacks. They also simulate different types of external attacks and check if the system is secure.*

*Test harnesses comprise small programs such as stubs and drivers to interact with the software being tested.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option D

Target 3 = Option C

Target 4 = Option E

Target 5 = Option B

**2. Using performance and monitoring tools**

During testing you can also use additional tools such as

* dynamic analysis tools
* performance testing tools
* monitoring tools

Dynamic analysis tools are used to analyze defects while the software code is executing. Developers use these tools during component testing and component integration testing to detect defects that are difficult to find during static testing.

For example, time dependencies lend themselves to dynamic testing, as does memory leakage handling, or boot-up processes.  
  
You can also use dynamic analysis tools to check websites for dead links.

Dynamic analysis tools can

* identify memory leaks
* identify pointer arithmetic
* detect time dependencies
* report the status of the software during code execution
* monitor the allocation, use, or deallocation of memory

Performance testing tools test the performance of a system under different load or usage patterns. Like test execution tools, performance testing tools use test scripts and data-driven testing, where variables that point to data in a data table are used to replace hard-coded inputs in test scripts.

You can also use performance testing tools to perform

**load testing**

In load testing, you check whether the system can cope with a certain number of transactions, loads, or usage patterns by pushing it beyond its normal and expected usage. You increase the load on the system gradually using a test driver so you can identify the usage pattern or the load that could make the system collapse. The output of the test is written to a log. After completion, you generate reports or graphs from the contents of the log to check the performance level.

**volume testing**

When you are checking volume, you test whether the system is able to handle a large amount of data. Take the case of a file that has many records in it. During volume testing, you check if the system can open and display all the records in the file.

**stress testing**

In stress testing, you check whether the system can handle more transactions than it is known to handle. You perform both load and volume testing. When you are checking load, you push the system to go beyond its normal and expected usage. You increase the load on the system gradually so you can identify the usage pattern or the load that can make the system collapse. This will help anticipate load issues after the system is deployed and help prevent them. For instance, when an HR system supports details of only 1,000 employees, entering information about the next employee might make the system collapse.

In performance testing, you verify how a system or a software application behaves under simulated usage conditions. For instance, when testing the performance of a web site, you check if the web site can handle the amount of traffic it is supposed to handle. Performance testing is best done at the system level. Ensuring the system is active, you replicate an end-user environment or user profiles. You send a number of test inputs to the system. The parameters that you set for the inputs may vary depending on the performance testing tool you use.

The tool then measures system characteristics such as response time and mean time between failure and sends you a report. If you find that the performance does not meet standards, you analyze where the problem is and how it can be improved.

Analyzing the output of performance testing tools takes time and skill. Therefore, it is a good idea to use specialists to carry out performance testing activities. This is especially so because the cost of some of these tools is quite high. Implementation and training costs increase the investment further. Using specialists will ensure that these tools are used optimally.

Performance testing tools are used to test an integrated group of systems, such as servers, databases, or an environment. Once functional, they need not be continually monitored.  
  
Performance testing tools can typically

* discover performance issues
* measure loads, average response times, and timing of specific transactions
* identify memory leaks
* generate graphs or charts of responses
* record problems such as locking, concurrency, excessive disk space, or system resources usage

Monitoring tools monitor the status and the performance of the systems in use. They are likely to be used after deployment. For instance, they may be used to recognize increasing demands on the system or abnormal behavior that requires alerting the administrator. If you use a tool to monitor your e-business websites, it will report any unusual behavior from visitors. If a monitoring tool detects a security attack then visitors can be automatically routed to another site, and site administrators receive an alert. You can prevent adverse consequences for the business such as negative publicity or financial loss.

Monitoring tools cover all aspects of IT infrastructure such as servers, databases, networks, performance, applications, and Internet usage. Because monitoring tools are used to monitor network traffic as well as the number of users on a network, they can help identify problems and alert the concerned people. They can also be used to log real-time information and historical data.

Apart from the testing tools covered, there are some other tools that are not specifically designed for testers or developers, but can still be used to support test activities. Examples of such tools are office productivity tools, database query tools, or communication tools, and backup or restore utilities.  
  
For instance, in the absence of a test management tools, you can use a spreadsheet to store data temporarily.

Question

Match the definitions with their different test execution and logging tools.

**Options:**

1. Monitoring tools
2. Dynamic analysis tools
3. Performance testing tools

**Targets:**

1. Analyze defects while the software code is executing
2. Check whether a system can stand up to a high volume of usage
3. Continuously check the status and the performance of the systems in use

Answer

*Dynamic analysis tools can analyze defects while the software code is executing. They can also perform other activities such as identifying memory leaks, detecting time dependencies, and monitoring the allocation of memory.*

*Performance testing tools check whether a system can stand up to a high volume of usage. These tools are also used to perform load and stress testing.*

*Monitoring tools continuously check the status and the performance of the systems in use. They also help identify problems that arise on a network connecting a large number of people.*

**Correct answer(s):**

Target 1 = Option B

Target 2 = Option C

Target 3 = Option A

Question

Match the required activities to the appropriate tools.

**Options:**

1. You are required to perform regression testing on an application that went through a version upgrade recently
2. After migrating a database to a new platform you want to detect any differences between the two versions
3. You've been assigned to perform component testing on an incomplete system
4. You want to test an application's defences against intrusion attacks

**Targets:**

1. Test harness
2. Test execution tools
3. Test comparators
4. Security testing tools

Answer

*A test harness can help you best during component or integration testing when parts of the system may be incomplete. It can measure coverage at the code level.*

*You should use test execution tools when you need to repeat a test performed earlier for a similar application.*

*Test comparators compare the expected results to the actual results and store the results either as part of the test case or as computed through a test oracle.*

*Security testing tools are used to test functions that detect security threats such as computer viruses, worms, or denial of service attacks.*

**Correct answer(s):**

Target 1 = Option C

Target 2 = Option A

Target 3 = Option B

Target 4 = Option D

Question

Match the definitions with their different test execution and logging tools.

**Options:**

1. You are testing a software application on the Internet. You find that the number of service requests directed at the application have increased drastically in the past hour.
2. You have to perform component testing on a software to check for pointer errors.
3. You want to detect any components that remain unexercised during a test.
4. You want to verify the working of a system under a simulated usage environment.

**Targets:**

1. Performance testing tools
2. Monitoring tools
3. Dynamic analysis tools
4. Coverage measurement tools

Answer

*Performance testing tools are used to test an integrated group of systems, such as servers, databases, or an environment under simulated usage conditions.*

*You use monitoring tools to recognize any abnormal increase in demand on a system that is deployed online.*

*You perform dynamic testing typically during component and integration testing to detect memory leaks, pointer errors, and time dependencies.*

*You use a coverage measurement tool to exercise these unexercised coverage items by identifying test inputs that can perform the exercise.*

**Correct answer(s):**

Target 1 = Option D

Target 2 = Option A

Target 3 = Option B

Target 4 = Option C

**3. Summary**

Test tools provide tool support for testing activities such as test execution and logging, and performance and monitoring.   
  
Test execution tools, test harnesses, test comparators, coverage measurement tools, and security tools help you in executing and logging test cases. Dynamic analysis tools, performance testing tools, and monitoring tools help you in activities such as analyzing code and monitoring performance.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#top)

Introducing a Tool into an Organization

Learning Objective

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

* *recognize when it is appropriate to introduce a test tool into an organization*

**1. Appropriate introduction of a test tool**

When an organization or project team decides to buy a tool to complete a task, they should study the advantages and disadvantages of buying the tool.

The factors you should keep in mind before buying the tool are

* maturity of the internal test processes used
* availability of alternate solutions
* constraints faced and requirements specified
* availability of tools that meet these requirements
* detailed evaluation/proof of concept
* cost of the selected tool

Before buying a test tool, you should first identify the strengths and weaknesses of the test organization. You should introduce the tool only if it can support the established test processes. To evaluate whether your organization is ready for tools, you can use the Test Process Improvement (TPI) model or the Capability Maturity Model Integration (CMMI) model. These models are software process improvement assessment models that can assess the maturity of an organization and provide guidelines for improving the test process.

It is a good practice to consider alternative solutions before investing in a tool. Trying out alternative solutions can sometimes be more beneficial for a test process than using a test tool or automating an old test pack.  
  
For example, your organization may require version management of its products. Instead of purchasing a configuration management test tool , you could look at alternative options, such as purchasing a cheaper open source tool, or using internal employees to develop this tool.

After you've ensured that there are no alternative solutions available, you should consider the constraints and requirements of the tool.  
  
The constraints could be financial or technical. The requirements could be about factors such as training needs or tool vendors' specifications. The requirements should be drawn up formally, prioritized and approved by the key stakeholders.

If you don't perform a detailed analysis of the need for the tool and specify the requirements and constraints, you could encounter problems such as delays, unnecessary expenditure, or inadequately equipped tools.  
  
For example, you may require a tool that can be remotely accessed. The tool you adjudged to be the best after a series of demonstrations might not be deployable over the Internet. Purchasing the tool without understanding its constraints and your own requirements would result in owning an inadequately equipped tool.

After specifying the requirements, next you should check if the available tools are capable of overcoming the constraints and satisfying the requirements. If they are, shortlist them for purchase.

After you decide to purchase a tool, you can approach tool vendors, attend tool exhibitions, and search for tools on the Internet.

When discussing with vendors, you should ensure that you provide them with a list of the constraints and the requirements so they are clear about your requirements.

After you decide to buy a tool, it is important that you evaluate both the tool vendor and the tool. This evaluation is called a proof of concept. When you've shortlisted only one tool, you can combine the pilot project, which is the first use trial project of the tool with the proof of concept, which is the detailed evaluation of the tool before purchase.

During evaluation, you should evaluate the tool in the same test environment where the tool will be used. This ensures that you don't need to reconfigure the tool. For example, some static analysis tools don't support all versions of all programming languages, so you should test a static analysis tool in the programming language version you intend to use.

After completing the evaluation, you should again assess the tools against the requirements. If there are any additional features, you can note them as potential future requirements.

After evaluating the tool, you should negotiate the cost of the tool with its vendor. During the negotiation, you should discuss the purchase price, consultancy costs, training and implementation costs, and annual license fees. You should also negotiate the costs regarding a pilot run of the tool and the costs associated with a large scale implementation of the tool.

After negotiating the costs, you should conduct a pilot project to test the tool. A pilot project helps you

**gain knowledge**

You can gain knowledge through a pilot project when you test the tool and identify what the benefits and shortcomings of the tool are.

**assess compatibility**

The pilot project helps you assess if the test tool is compatible with the existing processes.

**decide on process modifications**

During the pilot, you can determine what processes and practices you would need to modify to accommodate the tool.

**decide on how to ensure people make optimum use of the tool**

The pilot project helps you identify how people can make optimum use of the tool so existing processes can be streamlined.

**evaluate the benefits of the tool**

During the pilot, you can evaluate if the benefits the tool promises can be achieved at a reasonable price. This will help in deciding whether you want to purchase the tool.

**determine other details for using the tool**

The pilot project will also help you determine other details for using the tool. For example, you can identify the templates required and guidelines needed to use the tool.

If you're satisfied with the results of the pilot project, you should buy the tool. To implement the purchased tool, you should

**adopt an incremental approach**

You should adopt an incremental approach by releasing the tool to the rest of the organization in phases. After a pilot, you can release the tool into areas where it is likely to be more useful.

**adapt the tool to the existing processes**

You should find the right balance for using the existing tools and practices with the new tool. You can do this by adapting the tool to the existing processes, old tools, and testware.

**use the test pilot data**

You can use the test pilot data to create user guidelines that are to be used by the testing team.

**train employees**

You should train employees on using the tool. This ensures that they are prepared to use the tool in a live project.

**create a database**

You should compile a database of issues faced and learned based on the findings of the pilot. This ensures that the team will always have solutions to problems that have been encountered.

Question

You have been asked to define the requirements of the new test tool. Explain why you should define the requirements.

**Options:**

1. To prevent delays at a later stage
2. To avoid incurring additional costs at a later stage
3. To assess the maturity of the organization
4. To train the staff on the use of the tool

Answer

***Option 1:****Correct. Delays in your schedule can be caused when you use a tool that is inappropriate for testing activities. These delays will be prevented by defining the requirements of the tool.*

***Option 2:****Correct. Defining the requirements of a tool would ensure that no additional costs are incurred because of incompatible or inadequate tool features at a later stage.*

***Option 3:****Incorrect. The maturity of the organization will be assessed only to help decide whether or not it is practical to introduce a tool into the organization.*

***Option 4:****Incorrect. The staff will require training on the use of the tool only after the tool has been purchased and installed.*

**Correct answer(s):**

1. To prevent delays at a later stage  
2. To avoid incurring additional costs at a later stage

Question

You are a five year old company with established processes. Pick the two most appropriate reasons as to why it would be reasonable to implement a tool in the given scenarios.

**Options:**

1. You've carried out a TPI assessment and find your organization ready for a tool
2. You've sent a list of the constraints and requirements to a vendor
3. You've performed a thorough analysis of the constraints and requirements of a tool and are ready to implement a tool
4. You find that there are alternative solutions to purchasing a tool in the market

Answer

***Option 1:****Correct. A TPI assessment can help establish the maturity of the organization before you consider implementing a test tool.*

***Option 2:****Incorrect. When you send a list of the constraints and requirements to a vendor, it only establishes the limitations and needs you have and implementing a tool is not appropriate in this situation.*

***Option 3:****Correct. Performing a thorough analysis of the constraints and requirements of a tool will ensure that there are no subsequent delays, or unnecessary expenditure.*

***Option 4:****Incorrect. If there are alternative solutions in the market, you should first check them before deciding to implement a tool. Sometimes, alternative solutions such as training the staff or improving test practices are more beneficial to purchasing a tool.*

**Correct answer(s):**

1. You've carried out a TPI assessment and find your organization ready for a tool  
3. You've performed a thorough analysis of the constraints and requirements of a tool and are ready to implement a tool

**2. Summary**

Before introducing a new tool, you should assess the maturity of the organization, look for alternative solutions, analyze constraints and requirements of the tool, meet vendors, check all available tools, and create a detailed proof of concept. You should also take into consideration the cost of setting up the tool.  
  
After installing the tool, you should perform a pilot project to evaluate whether the tool would accomplish what is required. During the pilot, you should also determine if any existing processes should be modified to complement the tool and if the tool should be adapted to suit your processes.  
  
To successfully implement the tool, the organization should not only release the tool in stages, but should also implement the findings of the pilot to ensure the resources use the tool in the most efficient way. The organization should also create user guidelines and train the resources on using the tool. Additionally, it should create a database that holds a record of the issues faced during the pilot.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#top)

Software Test Tool Classification and Implementation

Learning Objectives

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

* *recognize the appropriate test tools to implement at each stage of the testing process*
* *recognize how to integrate tools into a testing process*

**1. Exercise Overview**

**2. Selecting test tool categories**

**3. Explaining the tools**

**4. Integrating the tools**

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_sftf_a08_it_enus/output/html/course_transcript.html#top)

© 2016 Skillsoft Ireland Limited