SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

School of Computing

Department of Computer Science and Engineering

and

Department of Information Technology

SCS1608 - Software Quality Assurance and Testing

UNIT - V

UNIT – V

TAXANOMY TO TESTING TOOLS

A wide variety of software testing tools are available to cater to the different types of software, different programming languages, and to carry out different types of testing. These testing tools can be broadly divided into the following categories:

- Functional/Regression testing tools
- Source code testing tools
- Performance testing tools
- Java testing tools
- Embedded software testing tools
- Network protocol testing tools
- Configuration management/bug tracking tools
- Testing management tools

Functional/Regression Testing Tools:

These tools are used to test the application software and web applications such as web sites. As majority of the applications involve Graphical User Interface (GUI), the tools test the GUI objects and functionality automatically. These tools carry out black box testing. Client/Server applications, Enterprise Resource Planning (EPvP) software packages such as SAP,Customer Relations Management (CRM) software packages such as Siebel, web sites etc. can be tested for functionality using these tools. Whenever a change is made to the software, the software needs to be retested and hence these tools are also called regression testing tools. Compuware's QACenter, Segue Software's SilkTest, IBM Rational's Robot, Mecury Interactive's WinRunner belong to this category.

Source Code Testing Tools:

These tools check the source code of the application software. The testing is white box testing and hence the implementation details are taken into consideration. A number of tools are available for checking line coverage, branch coverage, and path coverage. For instance, the profilers display the number of times each line is executed. The test engineer can study the profiler output to find out which portions of the code are not executed and then create test cases in such a way that the lines, which were not executed earlier, can be executed. Tools are also available to test whether the source code is compliant to the standard coding guidelines and to generate the metrics such as number of non-commented lines, number of commented lines, number of functions etc. Some tools check the portability of the code. For example, the code written in C is not portable if operating system dependent features are used. The 'lint' utility in Unix/Linux systems checks the portability of C code. Some application software source code testing tools are: AutomatedQA's AQtime, Parasoft's Insure++, and Telelogic's Logiscope.

Performance Testing Tools:

These tools are used to carry out performance testing or stress testing. These tools are very useful to test how the application works when multiple users access the application simultaneously. The application can be for example, a database or a web site. These tools simulate multiple users on a single machine and hence you do not need many machines and many test engineers to do the performance testing. AutoTester's AutoController, Compuware's QALoad, Mercury Interactive's LoadRunner, Segue Software's SilkPerformer, IBM Rational's Performance Tester, Apache JMeter belong to this category. Some other specialized tools in this category are Argogroup's MonitorMaster for testing mobile applications that use WML and XHTML, Short Messaging Service (SMS) and Multimedia Messaging Service (MMS); IBM Rational's Prevue-X and Prevue-ASCII for X-windows and ASCII terminal emulators.

Java Testing Tools:

As Java has become a popular programming language in recent years, a number of tools are available exclusively for testing Java applications. These tools are for testing applications written in Java programming language and for testing Java classes. Jemmy is an open source library to create automated tests for Java GUI applications. JMeter of Apache is another open source software to do performance testing. Parasoft's jtest is used for Java class testing.

Embedded Software Testing tools:

Testing embedded software is a very challenging task as the timing requirements for these applications are very stringent. In embedded systems, the code has to be optimized so that it occupies the minimum memory. IBM Rational Test Real Time is the widely used test tool in this category.

Network Protocol Testing tools:

As computer networks are becoming widespread, testing networking/communication software has attained lot of importance in recent years. A number of tools are available for testing networking and communication protocols. Many test instrumentation vendors such as Agilent Technologies, Rhode & Schwartz etc. supply protocol analyzers which generate the necessary protocols based on international standards such as ITU-T standards. netlQ's ANVL (Automated Network Validation Library) is to test routers and other networking products. This software generates packets in correct and incorrect formats to test the networking software. netlQ's Chariot is a network performance testing tool.

Configuration Management/Bug Tracking Tools:

In large software development projects, configuration management is a very important process. Also, when the test engineers report bugs, the managers have to track these bugs and ensure that all the bugs are removed. To facilitate this activity, good workflow management software is important. Many such tools, which are web-based are available. Bugzilla's Bugzilla is an open source defect tracking system. Samba's Jitterbug is a freeware defect tracking system. IBM Rational Software's Clear DDTS is a change request management software. GNU's GNATS is a freeware bug tracking and change management software. Segue Software's SilkRadar is an issue tracking and bug tracking software. Microsoft Excel is also used extensively for bug tracking. In Unix/Linux/ Solaris systems, utilities are available for version and release control of source code— these utilities are Source Code Control System (SCCS) and Revision Control System (RCS).

Software Testing Management Tools:

These tools help in managing process-oriented software testing. Using these tools, the QA manager can create a formal test plan, allocate resources, schedule unattended testing, track the status of various bugs/ activities etc. Auto Tester's AutoAdviser, Computer Software's QADirector, QIS's QCIT, Segue Software's SilkPlan Pro, IBM Rational Test Manager, Mercury Interactive's TestDirector are some such tools.

TEST AUTOMATION TOOL EVALUATION

1.Introduction

Success in any Test Automation (TA) effort lies in identifying the right tool for automation. A detailed analysis of various tools must be performed before selecting a tool. This requires a lot of effort and planning. The effort and learning obtained during tool evaluation will in turn help during the execution of the TA project.

Many companies fall prey to the sales executives of tool vendors, who show how easy it is to create scripts using their tool. It therefore becomes necessary to go into the details of the script thus created, to check the way by which the tool identifies and works with the product. In many cases, organizations have bought a tool license because it worked fine using record/playback. Eventually, they find out that the tool worked by identifying the product and its controls using coordinate positions, which is not a very reliable method, and the tool gets shelved. Hence a systematic approach must be taken to evaluate tools.

The entire process of tool evaluation can be broken down into three major phases:

- Requirements Gathering
- Tool Selection
- POC using selected tool

2.Requirements Gathering

During the requirements gathering phase of tool evaluation one has to list out the requirements for the automation tool. Some of the important questions you will need to ask would be:

- What problems will the tool solve?
- What technical capabilities will the tool need, to be compatible with your environment?

Some of the items that will help you guide your requirements list:

1 Compatibility issues

2 Tool audience

3 Management goals

4 Testing requirements

5 Technology

Compatibility Issues

Your testing tool will need to be compatible with:

- The operating systems your product supports
- The development environments used to create your product
- Third party software with which your product integrates
- The test management tool used (in order to be able to integrate with it); this will eliminate data redundancy and will give the advantage of managing all test related data at a single location
- The tool should also be version control friendly so that scripts created can be brought under source code control

Tool Audience

The skills of people involved in test automation and that of the people who use the automation scripts is another important criterion for the test tool evaluation process. The benefits obtained through automation boils down to how effectively the tool is being put to use.

Will your organization allow for staff training? Can your organization, within the implementation time, allow for the learning curve required to become comfortable with the tool?

Management goals

Typically, this will be based on the product roadmap and the goal for automation from the management perspective. Based on the product roadmap, the management might consider reviewing the time for which the tool has been in the market. They may also consider whether the tool will be upgraded periodically to support newer technologies. Any management will have a budget and will fit efforts within a given budget. So it is also important for the management to consider the licensing and maintenance cost of a tool and the additional hardware required for running the scripts.

Testing Requirements

What type of testing problems do you want the testing tool to address?

- Manual testing problems
- Time constraints when implementing small changes in the system
- Shorter regression testing timeframes
- Test data setup
- Defect tracking
- Increased test coverage
- Increased efficiency of the testing process

Technology

When considering technology requirements in the automation perspective, there are two views that must be taken:

- Technologies that should be supported by the tool
- Features required in the tool

The requirement could be for the tool to support all technologies that are used in the product and any new technology that is being planned to be implemented in the future. A list of all the technologies used in the product and the platforms/browsers on which the product is supported must be created as the technical requirements for the tool. For a tool to be able to automate testing of the product, the important criterion is the tool’s ability to identify, access and work with all controls used in the application. Hence it is important to create a list of all controls (standard, custom and third party) used in the application and check if the tool is able to identify and handle all the listed controls.

The technical stakeholders of the automation project will have a list of desirable features that they need in the tool. This list of desirable features must be created in consultation with the technical team. Apart from this, a list of all features supported by the tool must also be created so that all features of the tool are considered during evaluation. An example of few features that can be listed are the tool’s support for testing Graphical User Interface (GUI), Application Programming Interface (API) and Command Line Interface (CLI), support to extend tool using Open APIs like Win32 APIs, verification points supported in the tool, support of an efficient Recovery System and support for test report creation.

At the end of the requirements gathering phase, all necessary points to consider for selecting an automation tool are available. These points form the evaluation criteria for the tool.

1.Tools Selection

The second step in test automation tool evaluation is the selection of tools. While selecting tools it is important to remember that no single tool will satisfy all the requirements. The tool that meets most of the evaluation criteria should be chosen after discussion with stakeholders. Based on the tools limitations with respect to requirements, the automation activities must be planned.

All tools that meet most of the evaluation criteria can considered for evaluation. When many tools are found to satisfy the evaluation criteria, further analysis of tools should be done. Do a feature categorization by listing each tool according to the following features it provides:

- Mandatory features: These are the features that are essential to accomplish your goal in meeting your requirements within the constraints

- Desirable features: these are features that will distinguish the best tools from the others

- Irrelevant features: Features that are not important and will not provide any real benefit to your situation.

Rate these features and assess as many tools as possible to prepare a short list of a few tools. Contact the relevant vendors and possibly ask for an evaluation version to run your proof of concept (POC).

2.POC Using Selected Tool

The last phase of tool evaluation is doing a proof of concept. Though conceptually the tool appears to satisfy the evaluation criteria, it is necessary to try the tool for a few test scenarios / cases in the product. Every tool vendor provides an evaluation version of their tool for this purpose for a limited period of time. It is sufficient to use the evaluation version for the POC.

The scenarios chosen for POC are very important. They should be chosen in such a way that the scenarios cover most of the controls and a few common features present across the product. It should be a sample,

which, when automated should give the confidence that automation using the tool for the product will be successful. This will also help in finding available resources' competence in using the tool. In case the POC fails, the reasons for failure must be analyzed and documented. The next tool for POC should be selected and this can be an iterative process till you identify the tool that most likely,suits your requirements.

Conclusion

Tool evaluation is indeed a process in itself and requires a lot of research irrespective of who does the evaluation. The above listed process allows for you to make an informed decision with regards to the best tool to assist you with your software test automation effort.

LOAD RUNNER

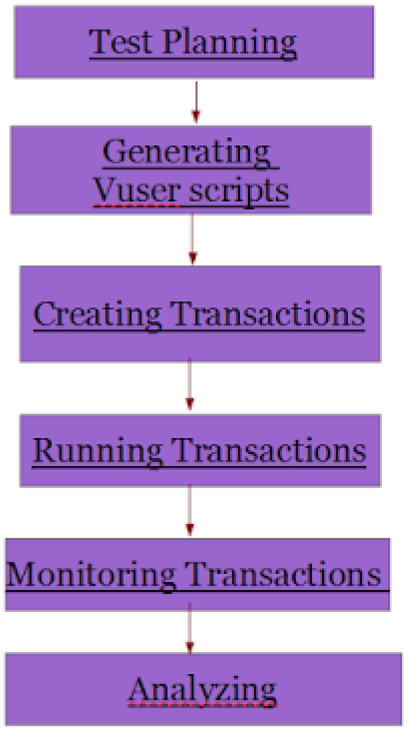
Why Load Runner??:

1. Load Runner: One of the best automated performance testing tool.
2. Uses ANSI C as the default programming language and other languages like Java and VB.
3. No need to install it on the server under test. It uses native monitors.
4. Supports all types of protocols (HTTP, FTP and SMTP).
5. Easy to analyze the results and creating scripts.

-

As we know Load Runner was acquired by **Hewlett-Packard** organization. One of the best and most used tool by many organizations. Though it's a paid tool, for perfect and easiness in use this tool is the best.

For every tool, there is a testing process to test an application. For the Load Runner , the best process to follow is :



The above model is the best model to follow for the performance test. Let's discuss much about the tool now. There are main components of Load Runner. They' are:

Components of Load Runner:

1) Vuser Generator
2) Controller
3)Analyzer

Let's discuss about each component briefly with related examples.

VUser Generator:

The first component and the basic component is "Vuser Generator". In Load Runner tool, humans are replaced by Vusers who are the replica of humans. More number of Vusers can be worked on a single work station with different scenarios. Load runner can accommodate hundreds or even thousands of Vusers with different scenarios.

With the help of Vuser script, users can perform the tests. User can record and playback the application for script generation.By modification or editing the scripts, user can create different scenarios for different Vusers. With this load test can be made simple and easy with one workstation.

Load runner supports scripting languages like ANSI C, VB Script, JAVA etc.. C and VB scripting are the most used ones in load runner. In recent versions of load runner JAVA scripting has been implemented, which can be widely used.

Controller:

In Load Runner 'Controller' is used to control the VUsers with single work station with different scenarios assigned to VUsers.

Analysis:

After the performance test the user can view the results of the test in graphs.More into the concepts of Load Runner. As we discussed, we start with 'Vuser Generator'.

Vuser scripts are created by Virtual User Generator with the recording of activities between client and server. It records the scripts. These scripts are used to emulate the steps of real human users. Using Vugen, we can also run the scripts for debugging.
VuGen can be used for recording in windows platforms. But, a recorded Vuser script can also be run on Unix platform.

Developing Vuser Script is a five step process:

- Record a Vuser script
- Vuser Script Enhancement – by adding the control statements and other functions
- Run time Settings Configuration
- Running of Vuser Script on Stand Alone machine – Verify that the script runs correctly
- Integration of Vuser Script – into a LoadRunner scenario or Performance Center or Tuning module session or Business process monitor profile etc.

WINRUNNER

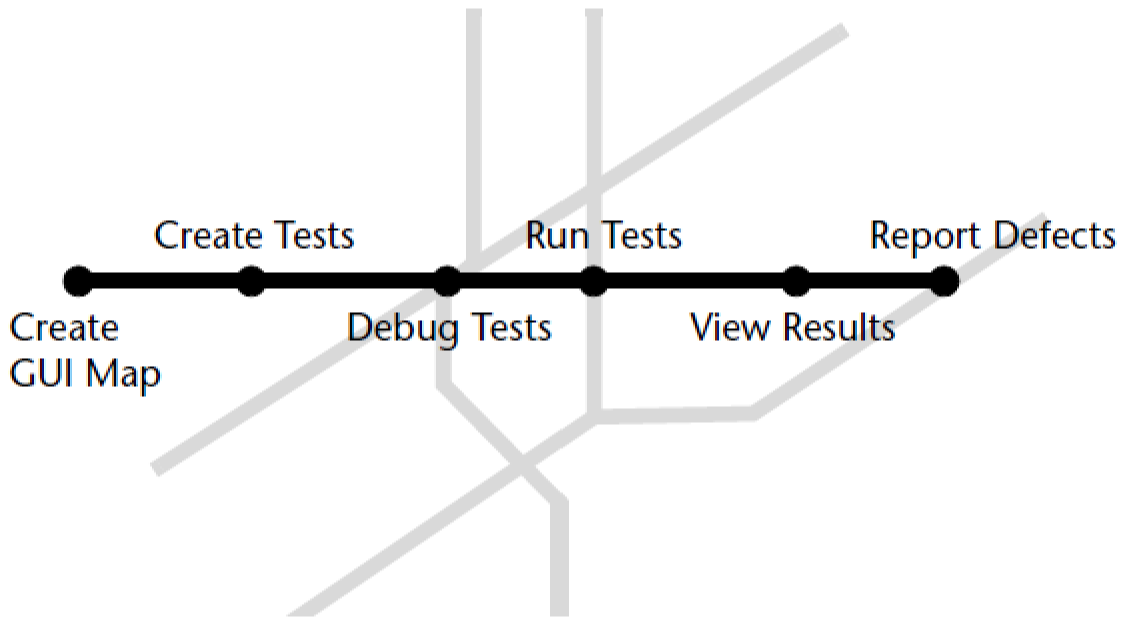
WinRunner is a testing tool to do functional/regression testing. Using WinRunner, you can record GUI operations. While recording, WinRunner automatically creates a test script. This test script can be run automatically later on for carrying out unattended testing. The important aspects of WinRunner are:

- You can do functional/regression testing of a variety of application software written in programming languages such as PowerBuilder, Visual Basic, C/C++, and Java. You can also carry out the testing on ERP/CRM software packages.
- You can do the testing in all flavors of Windows operating systems and different browser environments such as Internet Explorer and Netscape Navigator.

- You can record the GUI operations in the 'record' mode. WinRunner automatically creates a tesi script. This test can be modified if required and can be executed later on in unattend*aA*mode. The recovery manager enables the application to be brought to a known state in case there is a problem during the unattended testing. Rapid Test Script Wizard creates the test scripts automatically.
- You can add checkpoints to compare actual and expected results. The checkpoints can be GUI checkpoints, bitmap checkpoints and web links.
- It provides a facility for synchronization of test cases.
- Data Driver Wizard provides the facility to convert a recorded test into a data driven test. So, you c*i replace data with variables within a test script. For example, you can test a login process by taking the input for username and password fields from a database.
- Database checkpoints are used to verify data in a database during automated testing. The records that are inserted, deleted, modified, or updated will be highlighted so that you can ensure database integrity and transaction accuracy.
- The Virtual Object Wizard of WinRunner is used to teach WinRunner to recognize, record, and replay custom objects.
- The reporting tools provide the facility to generate automatically the test reports and analyze the defects.
- WinRunner can be integrated with the testing management tool TestDirector to automate many of the activities in the testing process

Testing with WinRunner involves six main stages:

Testing with *WinRunner* involves six main stages:



Create the GUI Map

The first stage is to create the GUI map so WinRunner can recognize the GUI objects in the application being tested. Use the RapidTest Script wizard to review the user interface of your application and systematically add descriptions of every GUI object to the GUI map. Alternatively, you can add descriptions of individual objects to the GUI map by clicking objects while recording a test.

Note that when you work in *GUI Map per Test* mode, you can skip this step.

Create Tests

Next, you create test scripts by recording, programming, or a combination of both. While recording tests, insert checkpoints where you want to check the response of the application being tested. You can insert checkpoints that check GUI objects, bitmaps, and databases. During this process, WinRunner captures data and saves it as expected results—the expected response of the application being tested.

Debug Tests

You run tests in Debug mode to make sure they run smoothly. You can set breakpoints, monitor variables, and control how tests are run to identify and isolate defects. Test results are saved in the debug folder, which you can discard once you've finished debugging the test.

When WinRunner runs a test, it checks each script line for basic syntax errors, like incorrect syntax or missing elements in ***If***, ***While***, ***Switch***, and ***For*** statements. You can use the Syntax Check options ***Tools Syntax Check) to check for these types of syntax errors before running your test.***

Run Tests

You run tests in Verify mode to test your application. Each time WinRunner encounters a checkpoint in the test script, it compares the current data of the application being tested to the expected data captured earlier. If any mismatches are found, WinRunner captures them as actual results.

View Results

You determine the success or failure of the tests. Following each test run, WinRunner displays the results in a report. The report details all the major events that occurred during the run, such as checkpoints, error messages, system messages, or user messages.

If mismatches are detected at checkpoints during the test run, you can view the expected results and the actual results from the Test Results window. In cases of bitmap mismatches, you can also view a bitmap that displays only the difference between the expected and actual results.

You can view your results in the standard WinRunner report view or in the Unified report view. The WinRunner report view displays the test results in a Windows-style viewer. The Unified report view displays the results in an HTML-style viewer (identical to the style used for QuickTest Professional testresults).

Report Defects

If a test run fails due to a defect in the application being tested, you can report information about the defect directly from the Test Results window.

This information is sent via e-mail to the quality assurance manager, who tracks the defect until it is fixed.

You can also insert **tddb_add_defect** statements to your test script that instruct WinRunner to add a defect to a TestDirector project based on conditions you define in your test script.

Rational Funtional Testing Tool

Rational Functional Tester is a tool for automated testing of software applications from the Rational Software division of IBM. It allows users to create tests that mimic the actions and assessments of a human tester.[1] It is primarily used by Software Quality Assurance teams to perform automated regression testing.

Rational Functional Tester is a software test automation tool used by quality assurance teams to perform automated regression testing. Testers create scripts by using a test recorder which captures a user's actions against their application under test. The recording mechanism creates a test script from the actions. The test script is produced as either a Java or Visual Basic.net application, and with the release of version 8.1, is represented as series of screen shots that form a visual storyboard. Testers can edit the script using standard commands and syntax of these languages, or by acting against the screen shots in the storyboard. Test scripts can then be executed by Rational Functional Tester to validate application functionality. Typically, test scripts are run in a batch mode where several scripts are grouped together and run unattended.

During the recording phase, the user may introduce verification points, which capture an expected system state, such as a specific value in a field, or a given property of an object, such as enabled or disabled. During playback, any discrepancies between the baseline captured during recording and the actual result achieved during playback are noted in the Rational Functional Tester log. The tester can then review the log to determine if an actual software bug was discovered.

Key Technologies

Storyboard Testing

Introduced in version 8.1 of Rational Functional Tester, this technology enables testers to edit test scripts by acting against screen shots of the application.

Object

The Rational Functional Tester Object Map is the underlying technology used by Rational Functional Tester to find and act against the objects within an application. The Object Map is automatically created by the test recorder when tests are created and contains a list of properties used to identify objects during playback.

ScriptAssure

During playback, Rational Functional Tester uses the Object Map to find and act against the application interface. However, during development it is often the case that objects change between the time the script was recorded and when a script was executed. ScriptAssure technology enables Rational Functional Tester to ignore discrepancies between object definitions captured during recording and playback to ensure that test script execution runs uninterrupted. ScriptAssure sensitivity, which determines how big an object map discrepancy is acceptable, is set by the user.

Data Driven Testing

It is common for a single functional regression test to be executed multiple times with different data. To facilitate this, the test recorder can automatically parametrize data entry values, and store the data in a spreadsheet like data pool. This enables tester to add additional test data cases to the test data pool without having to modify any test code. This strategy increases test coverage and the value of a given functional test.

Dynamic Scripting Using Find API

Rational Functional Test script, Eclipse Integration uses Java as its scripting language. The Script is a .java file and has full access to the standard Java APIs or any other API exposed through other class libraries.

Apart from this RFT itself provides a rich API to help user further modify the script generated through the recorder. RationalTestScript class that is the base class for any TestScript provides a find API that can be used to find the control based on the given properties.

Domains supported

(list is made based on the information for v 8.5, see here)

HTML Support: Mozilla Firefox, Internet Explorer, Google Chrome

Java

Abstract Window Toolkit (AWT) controls

Standard Widget Toolkit (SWT) controls

Swing or Java Foundation Class (JFC) controls

Eclipse (32-bit and 64-bit)

Dojo

WinForm and Windows Presentation Framework based .NET applications

ARM

Ajax

SAP WebDynPro

SAP - SAPGUI

Siebel

Silverlight

GEF

Flex

PowerBuilder

Visual Basic

Adobe PDF

Functional Tester Extensions for Terminal-based applications

SILK TEST

Introduction

Silk Test is a tool specifically designed for doing **REGRESSION AND FUNCTIONALITY**testing. It is developed by Segue Software Inc.

Silk Test is the industry's leading functional testing product for e-business applications, whether Window based, Web, Java, or traditional client/server-based. Silk Test also offers test planning, management, direct database access and validation, the flexible and robust 4Test scripting language, a built in recovery system for unattended testing, and the ability to test across multiple platforms, browsers and technologies.

You have two ways to create automated tests using silktest:

1. Use the **Record Testcase command** to record actions and verification steps as you navigate through the application.

2. Write the testcase manually using the **Visual 4Test scripting language**.

1. **Record Testcase**

The Record / Testcase command is used to record actions and verification steps as you navigate through the application. Tests are recorded in an object-oriented language **called Visual 4Test**. The recorded testreads like a logical trace of all of the steps that were completed by the user. The Silk Test point and click verification system allows you to record the verification step by selecting from a list of properties that are appropriate for the type of object being tested. For example, you can verify the text is stored in a text field.

2. **Write the Testcase manually**

We can write tests that are capable of accomplishing many variations on a test. The key here is re-use. A test case can be designed to take parameters including input data and expected results. This "data-driven" testcase is really an instance of a class of test cases that performs certain steps to drive and verify the application-under-test. Each instance varies by the data that it carries.

Since far fewer tests are written with this approach, changes in the GUI will result in reduced effort in updating tests. A data-driven test design also allows for the externalization of testcase data and makes it possible to divide the responsibilities for developing testing requirements and for developing test automation. For example, it may be that a group of domain experts create the Testplan Detail while another group of test engineers develop tests to satisfy those requirements.

In a script file, an automated testcase ideally addresses one test requirement. Specifically, a 4Test function that begins with the test case keyword and contains a sequence of 4Test statements. It drives an application to the state to be tested, verifies that the application works as expected, and returns the application to its base state.

A script file is a file that contains one or more related testcases. A script file has a .t extension, such as find .t

TESTING TOOLS FOR JAVA

1. Arquillian

Arquillian is a highly innovative and extendible testing platform for JVM that allows developers to easily create automated integration, functional and acceptance tests for Java. Arquillian allows you to run test in the run-time so you don't have to manage the run-time from the test (or the build). Arquillian can be used to manage the life cycle of the container (or containers),bundling test cases, dependent classes and resources. It is also capable of deploying archive into containers and execute tests in the containers and capture results and create reports.

Arquillian integrates with familiar testing frameworks such as JUnit 4, TestNG 5 and allows tests to be launched using existing IDE, and because of its modular design it is capable of running Ant and Maven test plugins.

2. JTest

JTest also known as ‘Parasoft JTest’ is an automated Java software testing and static analysis software made by Parasoft. JTest includes functionality for Unit test-case generation and execution, static code analysis, data flow static analysis, and metrics analysis, regression testing, run-time error detection.

There are also features that allow you to peer code review process automation and run-time error detection for e.g.: Race conditions, exceptions, resource and memory leaks, security attack vulnerabilities.

3. The Grinder

‘The Grinder’ is a Java load testing framework that was designed to make sure it was easy to run and distributed test’s using many load injector machines. The Grinder can Load test on anything that has a Java API. This includes HTTP web servers, SOAP and REST web services, and application servers and including custom protocols and the test scripts are written in the powerful Jython and Clojure languages. The GUI console for The Grinder allows you to have multiple load injectors to be monitored and controlled and Automatic management of client connections and cookies, SSL, Proxy aware and Connection throttling.

It is freely available under a BSD-style open-source license. You can find out more on their website.

4. TestNG

TestNG is a testing framework designed for the Java programming language and inspired by JUnit and NUnit. TestNG was primarily designed to cover a wider range of test categories such as unit, functional, end-to-end, integration, etc. It also introduced some new functionality that make it more powerful and easier to use, such as: Annotations, Running tests in big thread pools with various policies available, code testing in a multi thread safe, flexible test configurations, data-driven testing support for parameters, and more.

TestNG is supported by a variety of tools and plug-ins such as Eclipse, IDEA, Maven, etc

5. JUnit

JUnit is a unit testing framework designed for the Java programming language. JUnit has played an important role in the development of test-driven development frameworks. It is one of a family of unit testing frameworks which is collectively known as the xUnit that originated with SUnit.

JUnit is linked as a JAR at compile-time and can be used to write repeatable tests

6. JWalk

JWalk is designed as a unit testing toolkit for the Java programming language. It has been designed to support a testing paradigm called Lazy Systematic Unit Testing. The JWalkTester tool performs any tests of any compiled Java class, supplied by a programmer. It is capable of testing conformance to a lazy specification, by static and dynamic analysis, and from hints by the programmer behind the code.

7. Mockito

Mockito is designed as a open source testing framework for Java which is available under a MIT License. Mockito allows programmers to create and test double objects (mock objects) in automated unit tests for the purpose of Test-driven Development (TDD) or Behavior Driven Development (BDD).

8. Powermock

PowerMock is a Java Framework for unit testing of source code and It runs as an extension of other Mocking frameworks like Mockito or EasyMock but comes with more powerful capabilities. PowerMock utilizes a custom classloader and bytecode manipulator to enable mocking of static methods, removal of static initializes, constructors, final classes and methods and private methods. It as been primarily designed to extend the existing API's with a small number of methods and annotations to enable the extra features.

It is available under an open source Apache License 2..

9. JMeter

JMeter is a software that can perform load test, performance-oriented business (functional) test, regression test, etc., on different protocols or technologies.

Stefano Mazzocchi of the Apache Software Foundation was the original developer of JMeter. He wrote it primarily to test the performance of Apache JServ (now called as Apache Tomcat project). Apache later redesigned JMeter to enhance the GUI and to add functional testing capabilities.

JMeter is a Java desktop application with a graphical interface that uses the Swing graphical API. It can therefore run on any environment / workstation that accepts a Java virtual machine, for example − Windows, Linux, Mac, etc.

The protocols supported by JMeter are −

- Web − HTTP, HTTPS sites 'web 1.0' web 2.0 (ajax, flex and flex-ws-amf)
- Web Services − SOAP / XML-RPC
- Database via JDBC drivers
- Directory − LDAP
- Messaging Oriented service via JMS
- Service − POP3, IMAP, SMTP
- FTP Service

JMeter Features

Following are some of the features of JMeter −

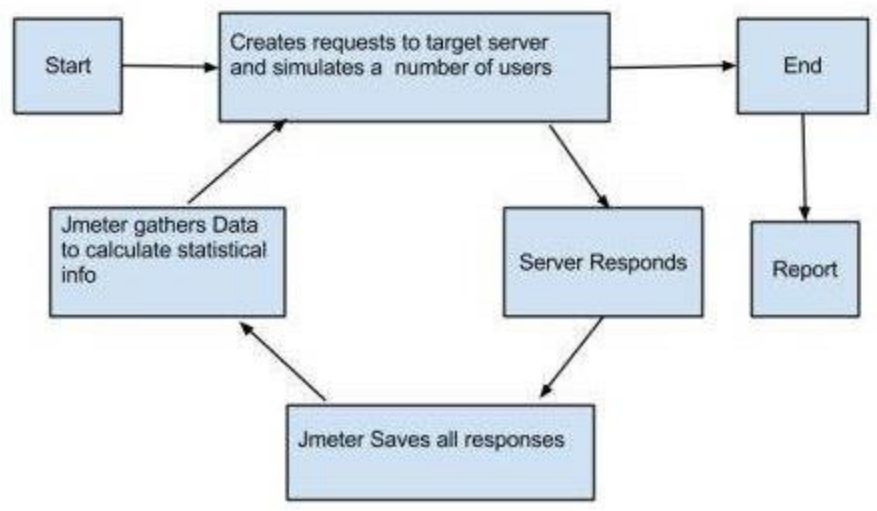
- Being an open source software, it is freely available.
- It has a simple and intuitive GUI.
- JMeter can conduct load and performance test for many different server types − Web - HTTP, HTTPS, SOAP, Database via JDBC, LDAP, JMS, Mail - POP3, etc.
- It is a platform-independent tool. On Linux/Unix, JMeter can be invoked by clicking on JMeter shell script. On Windows, it can be invoked by starting the jmeter.bat file.

- It has full Swing and lightweight component support (precompiled JAR uses packages javax.swing.*).
- JMeter store its test plans in XML format. This means you can generate a test plan using a text editor.
- Its full multi-threading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.
- It is highly extensible.
- It can also be used to perform automated and functional testing of the applications.

How JMeter Works?

JMeter simulates a group of users sending requests to a target server, and returns statistics that show the performance/functionality of the target server/application via tables, graphs, etc.

Take a look at the following figure that depicts how JMeter works −


Start
Creates requests to target server and simulates a number of users
End
Jmeter gathers Data to calculate statistical info
Server Responds
Report
Jmeter Saves all responses

What is Junit?

Junit is widely used testing framework along with Java Programming Language. You can use this automation framework for both unit testing and UI testing.It helps us define the flow of execution of our code with different Annotations. Junit is built on idea of "first testing and then coding" which helps us to increase productivity of test cases and stability of the code.

Important Features of Junit Testing -

1. It is open source testing framework allowing users to write and run test cases effectively.
2. Provides various types of annotations to identify test methods.
3. Provides different Types of Assertions to verify the results of test case execution.
4. It also gives test runners for running tests effectively.
5. It is very simple and hence saves time.
6. It provides ways to organize your test cases in form of test suits.
7. It gives test case results in simple and elegant way.
8. You can integrate Jnuit with Eclipse ,Android Studio,Maven & Ant

Following are the JUnit extensions −

- Cactus
- JWebUnit
- XMLUnit
- MockObject

Cactus

Cactus is a simple test framework for unit testing server-side java code (Servlets, EJBs, Tag Libs, Filters). The intent of Cactus is to lower the cost of writing tests for server-side code. It uses JUnit and extends it. Cactus implements an in-container strategy that executes the tests inside a container.

Cactus ecosystem is made of several components −

- **Cactus Framework** is the heart of Cactus. It is the engine that provides the API to write Cactus tests.

- **Cactus Integration Modules** are front-ends and frameworks that provide easy ways of using the Cactus Framework (Ant scripts, Eclipse plugin, and Maven plugin).