RECOVERY TESTING

Recovery testing is a type of non-functional testing technique performed in order to determine how quickly the system can recover after it has gone through system crash or hardware failure. Recovery testing is the forced failure of the software to verify if the recovery is successful.

TESTING CRITERIA OF RECOVERY TESTING PROCESS:

Software or system is expected to recover from failures when −

- There is a loss of wireless signal
- The external device doesn't respond
- The power supply fails abruptly
- The system fails due to physical conditions
- The server fails to respond properly
- The system could not detect DLL file
- Other network issues are detected

Executing a recovery test in six steps:

A tester must follow these six crucial steps while performing a recovery test −

- **Step 1 − Recovery analysis** – It is important that the system under test can allocate extra resources like multiple CPUs, and servers. This helps in understanding how each recovery-based change affects the working structure of the system. Besides reporting possible failures, the testers must also analyze the impact and the severity of the impact of such failures.
- **Step 2 − Preparing Test Plan** – Once recovery analysis results are recorded, the testing team prepares the test plan.

- **Step 3 − Preparing Test Environment** – This step involves evaluating the results from the recovery analysis process and designing the test environment.
- **Step 4 − Back-up Maintenance** – To recover any possible data loss during the test, the testing team performs a backup of all information related to the system, software, and database. If essential data is there, it should also be back-up and stored in multiple locations for safety.
- **Step 5 − Allocating recovery personnel** – Since this process is divided into steps, special recovery personnel is allocated for each step.
- **Step 6 − Documentation** – Each step performed before and during the test is analyzed carefully when encountered a failure.

Factors to consider while conducting a recovery testing:

Make sure to follow these steps while performing a recovery test −

- You must create a testbed similar to the real conditions of deployment. You must ensure all conditions like interface, firmware, protocol, software, and hardware are close as possible to the real condition.
- Run exhaustive testing regardless of how much it costs. A complete identical configuration and check are necessary.
- Try to run the test on the actual hardware you intend to restore the program after the test.
- For the backup system, try to get hardware with a similar size to the drive from where you have taken the backup.
- Try to discard obsolete technology and use the latest hardware/firmware to avoid compatibility issues. If such issues arrive, you can create a virtual machine for the hardware with the same disk sizes and configuration.
- Creating an online backup system is a great way to avoid exposure to media problems. Most online backup systems are reliable. However, you should

also check their restore ability, retrieval functionality, encryption level, and overall security.

Examples of Recovery Testing

It's quite common to experience system failure, but it is also crucial to ensure the system can recover with ease causing minimum to no damage to the user's essential data.

Here are some examples of how a recovery testing contributes to a system performance −

Example 1

Let's say you are running multiple sessions in your browser and the network goes off or the system turns off due to power failure or any other condition. In recovery testing, testers test your browser under such a scenario and ensure the browser recovers from the failure and all your previous sessions are restores completely after a system restart.

Example 2

If you are streaming a movie over a network and suddenly the software crashed or you clicked the close button mistakenly. Will you be able to resume the movie from the place you left off? In recovery testing, testers simulate a software failure by unplugging the system power and plug it again to ensure the app recovers and resumes receiving data from the point of failure.

Example 3

Assume you are sitting inside a café and downloading a movie from the café's Wi-Fi. Suddenly you move to a non-Wi-Fi zone and the downloading stops in the middle. Now, if your move back to the Wi-Fi zone will the download resumes

from the same point or it you have to download the file again? Testers simulate the whole process to check the recovery rate of the software.

ARESTORATION STRATEGY

A restoration strategy is planned by the recovery team to retrieve important code and data bringing all operations go back to their normal state. The strategies might differ based on the criticality of the system.

A mockup restoration strategy −

- Try to do at least a single backup or multiple if the stakes are high
- In case of multiple backups, try to store them at different places
- Choose between an online or offline backup or both if necessary
- Check the policy to ensure if you can conduct an automatic backup or do it manually
- It would help to have an independent restoration team or development team by your side
- Note − The cost of restoration strategy will increase if you go for multiple backups or hire an independent team.

POST-RESTORATION TESTING

Once all files and folders are restored, make sure to do the following activities −

- Count the files from the restored folders and make sure it matches with the backup folder.
- Rename the corrupted document folder.
- Check whether the restored files are working. Open them with applications that you use normally. Also, check if you can browse, update, and modify the data.

- Open all types of files such as documents, music, pictures, and videos including small and large ones.

CONFIGURATION TESTING

Configuration testing is the process of testing a system with each of the supported software and hardware configurations. The Execution area supports configuration testing by allowing reuse of the assigned tests.

This testing is done to determine the optimal configurations under which a system or an application can work fine without any bugs, issues or flaws in performance. So, the most efficacious configuration that will deliver the required performance characteristics is spotted with the help of this testing.

The second main reason for this testing is to verify the system's compatibility with the other software or equipment signified in the SRS (software requirement specification).

For Example, you will check how the application works with the combination of Windows 10 OS, Windows Server 2016 & MySQL database, then you will perform another test to check how the application works with the combination of Windows 10 OS, Windows Server 2016 and IBM DB2 database. And, so on until you test all the possible configurations.

Our testing would not only be restricted to the software, but it will also cover the hardware, where we will have to check each of the combinations of various hardware devices. Therefore, at times this testing is also referred as Hardware Configuration Testing.

An Example

Let's consider that your enterprise has developed a desktop application in C# language and this app is built on the .NET framework.

And this app is based on a 3-tier architecture which has three layers – front end (client), the application layer (server) and the database layer. Each of the layers will support certain platforms accordingly.

PREPARING FOR CONFIGURATION TEST

This testing has certain pre-requisites that need to be fulfilled before we execute the configuration tests.

Below are the prerequisites:

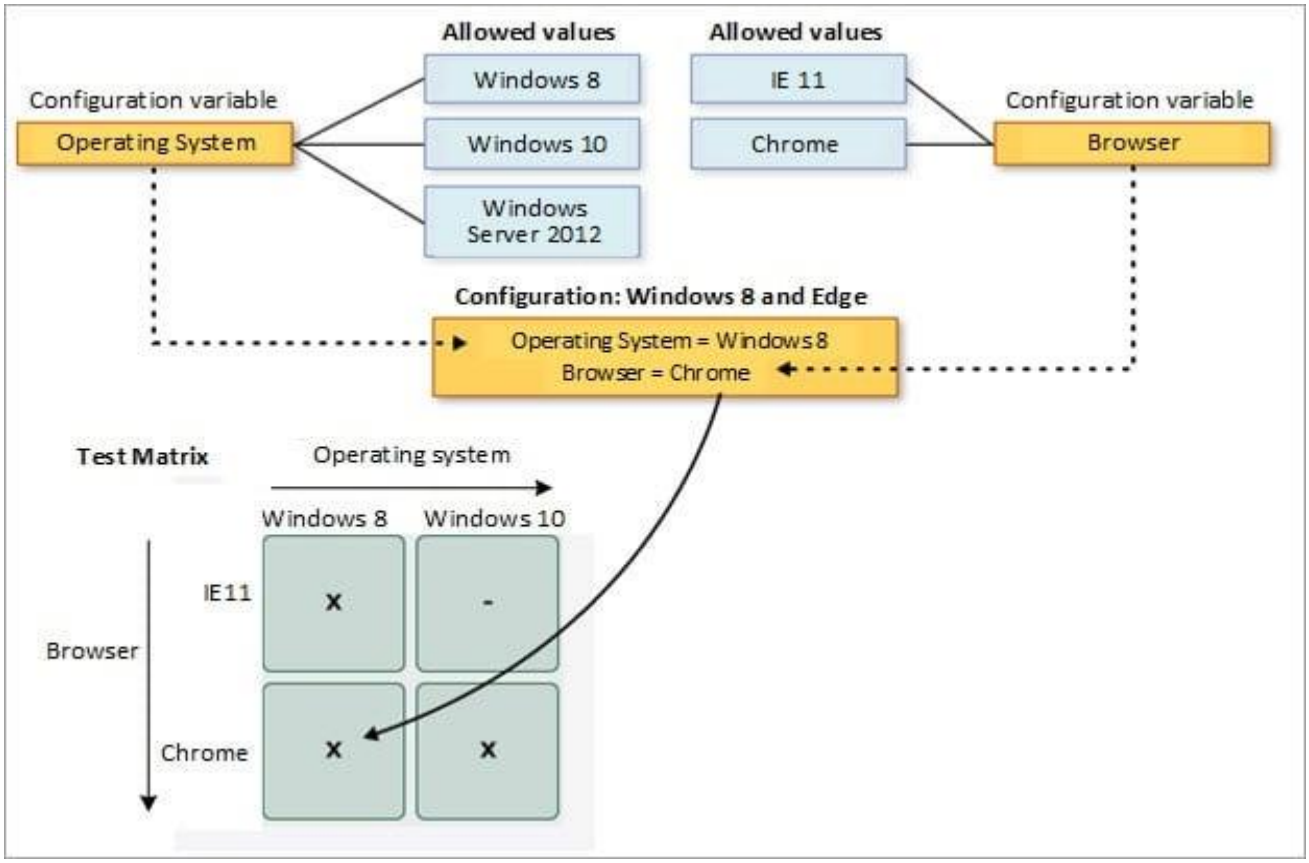
#1) PREPARING THE COVERAGE MATRIX

Owing to a huge number of possible hardware & software configurations, it becomes very time-consuming and almost impossible to test each of the configurations effectively.

For Instance, in the example that we discussed above, we will have a total of 3*3*3 i.e. 27 software configurations. And, let's suppose we have 5 different hard drives and 6 different memory sizes. Then, the count will go to 27*5*9 i.e. 810 configurations now.

This will keep on increasing if we add more components to the picture. So, it becomes crucial to do the planning for the software testing effort and identify clearly that which platforms will be supported.

Then, we need to come up with a coverage matrix that will hold the various combinations of the hardware & software configurations. Sometimes, this coverage matrix is also known as BCM (Basic Configuration Matrix).



The above figure shows a sample schematic matrix of configurations that you would wish to test.

#2) Prioritizing the Configurations

Once the configuration matrix is prepared, the next step is to prioritize the configurations.

This step is required because it is impossible to test the whole huge wide range of configurations. So, based on the client feedback, the most critical configurations are enlisted, and they are to be thoroughly tested first.

Once, we are done with the above two steps, we can go ahead with testing the various configurations based on their priority.

Microsoft VSTS – Configuration Testing Tool

Microsoft Visual Studio Team Services (VSTS) is a tool which greatly helps in testing your app under various configurations based on your test plan.

You should have a test plan to decide which tests you want to execute and on which configurations. You need to ensure that you have the right environment set up for the configurations you require. Once you have the matrix of the combinations you need to test it.

You can follow the below steps to perform this testing:

#1) Set up the configurations and create the variables. A variable is one of the components in your configuration.

For Example, there can be a variable 'Browser' which can have multiple values like Chrome, Firefox, IE10, etc.

#2) Assign the configurations to the test plans/test suites or individual test cases.

#3) Execute the tests against each configuration.

#4) Track the test results for each of the configurations.