# Software Re-engineering

# Software Re-engineering is a process of software development that is done to improve the maintainability of a software system. Software Re-Engineering is the examination and alteration of a system to reconstitute it in a new form.

# This process encompasses a combination of sub-processes like reverse engineering, forward engineering, reconstructing, etc.

# It positively affects software cost, quality, customer service, and delivery speed. In Software Re-engineering, we are improving the software to make it more efficient and effective.

# It is a process where the software’s design is changed and the source code is created from scratch. Sometimes software engineers notice that certain software product components need more upkeep than other components, necessitating their re-engineering.

# The need for software Re-engineering:

# Software re-engineering is an economical process for software development and quality enhancement of the product. This process enables us to identify the useless consumption of deployed resources and the constraints that are restricting the development process so that the development process could be made easier and cost-effective (time, financial, direct advantage, optimize the code, indirect benefits, etc.) and maintainable. The software reengineering is necessary for having-

# a) Boost up productivity: Software reengineering increase productivity by optimizing the code and database so that processing gets faster.

# b) Processes in continuity: The functionality of older software products can be still used while the testing or development of software.

# c) Improvement opportunity: Meanwhile the process of software reengineering, not only software qualities, features, and functionality but also your skills are refined, and new ideas hit your mind. This makes the developer’s mind accustomed to capturing new opportunities so that more and more new features can be developed.

# d) Reduction in risks: Instead of developing the software product from scratch or from the beginning stage, developers develop the product from its existing stage to enhance some specific features brought in concern by stakeholders or its users. Such kind of practice reduces the chances of fault fallibility.

# e) Saves time: As stated above, the product is developed from the existing stage rather than the beginning stage, so the time consumed in software engineering is lesser.

# f) Optimization: This process refines the system features, and functionalities and reduces the complexity of the product by consistent optimization as maximum as possible.

# Re-Engineering cost factors:

# The quality of the software is to be re-engineered.

# The tool support availability for engineering.

# The extent of the data conversion is required.

# The availability of expert staff for Re-engineering.

# Steps involved in Software Re-Engineering

# 

# 1. Inventory Analysis: Every software organization should have an inventory of all the applications.

# Inventory can be nothing more than a spreadsheet model containing information that provides a detailed description of every active application.

# By sorting this information according to business criticality, longevity, current maintainability, and other local important criteria, candidates for re-engineering appear.

# The resource can then be allocated to a candidate application for re-engineering work.

# 2. Document reconstructing: Documentation of a system either explains how it operates or how to use it.

# Documentation must be updated.

# It may not be necessary to fully document an application.

# The system is business-critical and must be fully re-documented.

# 3. Reverse Engineering: Reverse engineering is a process of design recovery. Reverse engineering tools extract data and architectural and procedural design information from an existing program.

# 

# 

# 4. Code Reconstructing:

# To accomplish code reconstruction, the source code is analyzed using a reconstructing tool. Violations of structured programming construct are noted and code is then reconstructed.

# The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced.

# 5. Data Restructuring:

# Data restructuring begins with a reverse engineering activity.

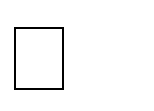
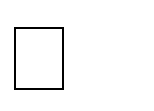
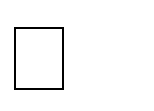
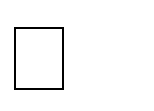
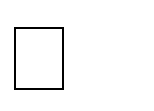
# The current data architecture is dissected, and the necessary data models are defined.

# Data objects and attributes are identified, and existing data structures are reviewed for quality.

# 6. Forward Engineering: Forward Engineering also called renovation or reclamation not only recovers design information from existing software but uses this information to alter or reconstitute the existing system to improve its overall quality.

# 

## Re-Engineering Process

* **Decide** what to re-engineer. Is it whole software or a part of it?
* **Perform** Reverse Engineering, in order to obtain specifications of existing software.
* **Restructure Program** if required. For example, changing function-oriented programs into object-oriented programs.
* **Re-structure data** as required.
* **Apply Forward engineering** concepts in order to get re-engineered software.

## Reverse Engineering

* It is a process to achieve system specification by thoroughly analysing, understanding the existing system. This process can be seen as reverse SDLC model, i.e. we try to get higher abstraction level by analysing lower abstraction levels.
* An existing system is previously implemented design, about which we know nothing. Designers then do reverse engineering by looking at the code and try to get the design. With design in hand, they try to conclude the specifications. Thus, going in reverse from code to system specification.

## Program Restructuring

* It is a process to re-structure and re-construct the existing software. It is all about re- arranging the source code, either in same programming language or from one programming language to a different one. Restructuring can have either source code- restructuring and data-restructuring or both.
* Re-structuring does not impact the functionality of the software but enhance reliability and maintainability. Program components, which cause errors very frequently can be changed, or updated with re-structuring.
* The dependability of software on obsolete hardware platform can be removed via re- structuring.

## Forward Engineering

* Forward engineering is a process of obtaining desired software from the specifications in hand which were brought down by means of reverse engineering. It assumes that there was some software engineering already done in the past.
* Forward engineering is same as software engineering process with only one difference – it is carried out always after reverse engineering.

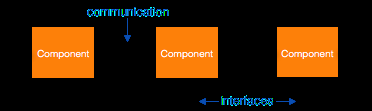


## Component reusability

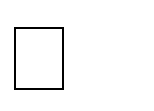
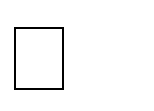
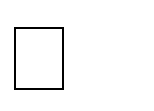
A component is a part of software program code, which executes an independent task in the system. It can be a small module or sub-system itself.

### Example

* The login procedures used on the web can be considered as components, printing system in software can be seen as a component of the software.
* Components have high cohesion of functionality and lower rate of coupling, i.e. they work independently and can perform tasks without depending on other modules.
* In OOP, the objects are designed are very specific to their concern and have fewer chances to be used in some other software.
* In modular programming, the modules are coded to perform specific tasks which can be used across number of other software programs.
* There is a whole new vertical, which is based on re-use of software component, and is known as Component Based Software Engineering (CBSE).



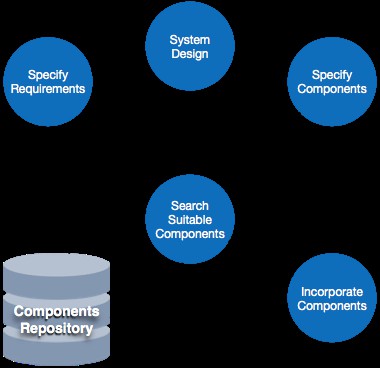
Re-use can be done at various levels

* **Application level** - Where an entire application is used as sub-system of new software.
* **Component level** - Where sub-system of an application is used.
* **Modules level** - Where functional modules are re-used.

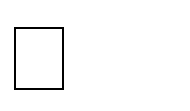
Software components provide interfaces, which can be used to establish communication among different components.

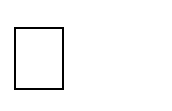
## Reuse Process

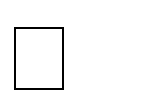
Two kinds of method that can be adopted: either by keeping requirements same and adjusting components or by keeping components same and modifying requirements.

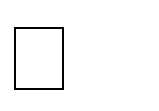


**Requirement Specification** - The functional and non-functional requirements are specified, which a software product must comply to, with the help of existing system, user input or both.

**Design** - This is also a standard SDLC process step, where requirements are defined in terms of software parlance. Basic architecture of system as a whole and its sub-systems are created.

**Specify Components** - By studying the software design, the designers segregate the entire system into smaller components or sub-systems. One complete software design turns into a collection of a huge set of components working together.

**Search Suitable Components** - The software component repository is referred by designers to search for the matching component, on the basis of functionality and intended software requirements.

**Incorporate Components** - All matched components are packed together to shape them as complete software.

# Software CASE Tools

CASE (Computer Aided Software Engineering) tools are set of software applications which are used to automate SDLC activities.

It means development and maintenance of software projects with help of various automated software tools.

CASE tools are used by software project managers, analysts and engineers to develop software system.

There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools.

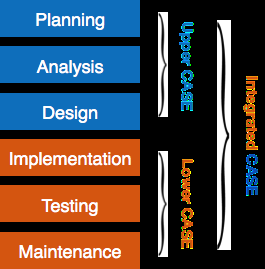
Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

## Components of CASE Tools

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

* **Central Repository:** CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as **data dictionary**.
* **Upper Case Tools** - Upper CASE tools are used in planning, analysis and design stages of SDLC.
* **Lower Case Tools** - Lower CASE tools are used in implementation, testing and maintenance.
* **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.



## Scope of Case Tools

The scope of CASE tools goes throughout the SDLC. Now we briefly go through various CASE tools:-

### Diagram tools

These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

### Process Modelling Tools

Process modelling is method to create software process model, which is used to develop the software. Process modelling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF composer.

### Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

### Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.

Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

### Analysis Tools

These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

### Design Tools

These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design.

### Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal with –

* + Version and revision management
  + Baseline configuration management
  + Change control management

CASE tools help in this by automatic tracking, version management and release management. For example, Fossil, Git, Accu REV.

### Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

### Programming Tools

These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

### Prototyping Tools

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.

Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.

### Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, and script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets.

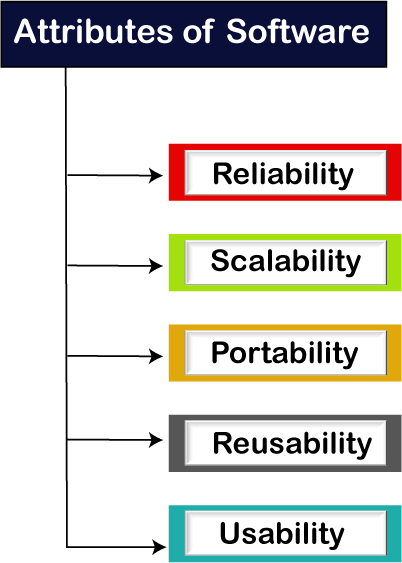
### Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.

### Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error Ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Centre.

# WHAT IS SOFTWARE TESTING

Software testing is a process of identifying the correctness of software by considering its all attributes (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects.

Software testing provides an independent view and objective of the software and gives surety of fitness of the software. It involves testing of all components under the required services to confirm that whether it is satisfying the specified requirements or not. The process is also providing the client with information about the quality of the software.

Testing is mandatory because it will be a dangerous situation if the software fails any of time due to lack of testing. So, without testing software cannot be deployed to the end user.



# Manual testing

* + The process of checking the functionality of an application as per the customer needs without taking any help of automation tools is known as manual testing. While performing the manual testing on any application, we do not need any specific knowledge of any testing tool, rather than have a proper understanding of the product so we can easily prepare the test document.
  + Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Test case reports are also generated manually.
  + Manual Testing is one of the most fundamental testing processes as it can find both visible and hidden defects of the software. The difference between expected output and output, given by the software, is defined as a defect. The developer fixes the defects and handes it to the tester for retesting.
* Manual testing is mandatory for every newly developed software before automated testing. This testing requires great efforts and time, but it gives the surety of bug-free software. Manual Testing requires knowledge of manual testing techniques but not of any automated testing tool.

# Why we need manual testing

* Whenever an application comes into the market, and it is unstable or having a bug or issues or creating a problem while end-users are using it.
* If we don't want to face these kinds of problems, we need to perform one round of testing to make the application bug free and stable and deliver a quality product to the client, because if the application is bug free, the end-user will use the application more conveniently.
* If the test engineer does manual testing, he/she can test the application as an end-user perspective and get more familiar with the product, which helps them to write the correct test cases of the application and give the quick feedback of the application.

**Manual testing can be further divided into three types of testing:**

# White-box testing

The white box testing is done by **Developer**, where they check every line of a code before giving it to the Test Engineer. Since the code is visible for the Developer during the testing, that's why it is also known as White box testing.

# Black box testing

The black box testing is done by the **Test Engineer**, where they can check the **functionality** of an application or the software **according to the customer/client's needs**. In this, the code is not visible while performing the testing; that's why it is known as black-box testing.

# Grey Box testing

Grey box testing is a **combination of white box and Black box testing**. It can be performed by a **person** who knew **both coding and testing**. And if the single person performs white box and black-box testing for the application, is known as Grey box testing.

# How to perform Manual Testing

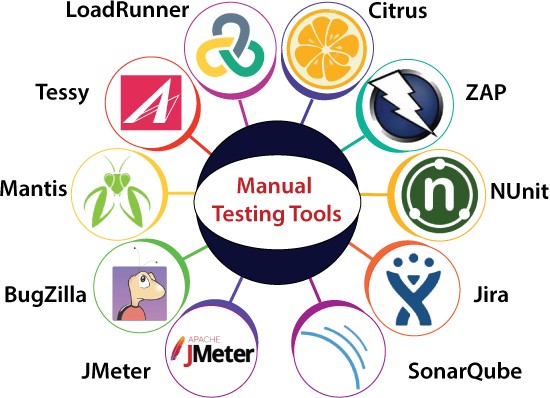
1. First, tester observes all documents related to software, to select testing areas.
2. Tester analyses requirement documents to cover all requirements stated by the customer.
3. Tester develops the test cases according to the requirement document.
4. All test cases are executed manually by using Black box testing and white box testing.
5. If bugs occurred then the testing team informs the development team.
6. The Development team fixes bugs and provides the software to the testing team for a retest.

# Advantages of Manual Testing

1. Manual testing can be done on all kinds of applications
2. It is preferable for short life cycle products
3. Newly designed test cases should be executed manually.
4. Application must be tested manually before it is automated
5. It is preferred in the projects where the requirements change frequently and for the products where the GUI changes constantly
6. It is cheaper in terms of initial investment compared to Automation testing
7. It requires less time and expense to begin productive manual testing
8. It allows tester to perform adhoc testing
9. There is no necessity to the tester to have knowledge on Automation Tools

# Disadvantages of Manual Testing

1. Manual Testing is time-consuming mainly while doing regression testing.
2. Manual testing is less reliable compared to automation testing because it is conducted by humans. So there will always be prone to errors and mistakes.
3. Expensive over automation testing in the long run.
4. It is not possible to reuse because this process can’t be recorded.



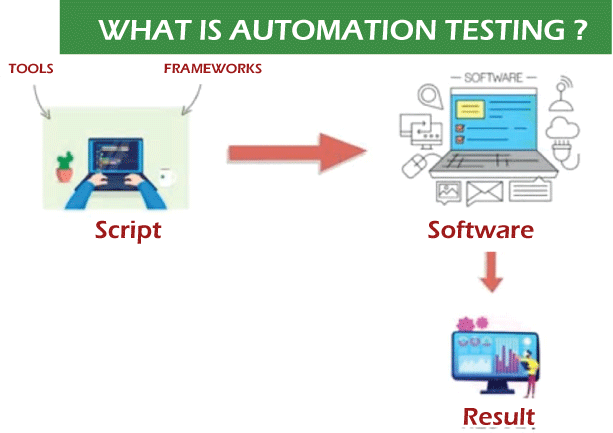
# Manual testing tools

In manual testing, different types of testing like unit, integration, security, performance, and bug tracking, we have various tools such as Jira, Bugzilla, Mantis, Zap, NUnit, Tessy, LoadRunner, Citrus, SonarQube, etc. available in the market. Some of the tools are open-source, and some are commercial.

# Automation Testing

Automation testing refers to the automatic testing of the software in which developer or tester writes the test script once with the help of testing tools and framework and run it on the software. The test script automatically test the software without human intervention and shows the result (either error, bugs are present or software is free from them).

* Automation testing needs manual effort when creating initial scripts, and further process is performed automatically to compare the actual testing result with expected results.
* In automation testing, the test automation engineer will write the test script or use the automation testing tools to execute the application. On the other hand, in manual testing, the test engineer will write the test cases and implement the software on the basis of written test cases.
* In test automation, the test engineer can execute repetitive tasks and other related tasks. In manual testing, it is a tedious process to implement the repetitive take again and again.
* In other words, we can say that the main concentration of test Automation is to change the manual human activity with systems or devices.
* The automation testing process is a time-saving process as it spends less time in exploratory testing and more time in keeping the test scripts whereas enhancing the complete test coverage.



# Advantages

* 1. **Reusability**

We can re-use the test scripts in automation testing, and we don't need to write the new test scripts again and again. And, we can also re-create the steps which are detailed as the earlier ones.

* 1. **Consistency**

As compared to manual testing, automation testing is more consistent and faster way than executing the regular monotonous tests that cannot be missed but may cause faults when tested manually.

* 1. **Running Tests 24/7**

In automation testing, we can start the testing process from anywhere in the world and anytime we want to. And even we can do that remotely if we don't have many approaches or the option to purchase them.

* 1. **Early Bug Detection**

We can easily detect the critical bugs in the software development process's initial phases by executing automation testing. It also helps us spend fewer working hours to fix these problems and reduce costs.

* 1. **Less Human Resources**

To implement the automation test script, we need a test automation engineer who can write the test scripts to automate our tests, rather than having several people who are repeatedly performing the tedious manual tests.

**MANUAL TESTING VS. AUTOMATION TESTING**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MANUAL TESTING** | | **AUTOMATION TESTING** | | | |
| Testing in which a human tester executes test cases | | Automation tools are used to execute the test cases | | | |
| In this testing, human resources are involved, that's why it is time- consuming | | It is much faster than the manual testing | | | |
| It is repetitive and error-prone | | Here automated tools are used that make it fast and accurate | | | |
| BVT (build verification testing) is time- consuming and tough in manual testing | | It's easy to build verification testing | | | |
| Instead of frameworks, this testing use checklist, guidelines, and stringent process for drafting test cases. | | | Frameworks like keyword, hybrid, and data drive to accelerate the automation process. | | | |
| It is best for usability, exploratory and adhoc testing | | It is widely used for performing testing, load testing and regression testing. | | | |
| The process turnaround time is higher than the automation testing process (one testing cycle takes lots of time) | | It completes a single round of testing within record time; therefore, a process turnaround time is much lower than a manual testing process. | | | |
| The main goal of manual testing is  user-friendliness or improved customer experience. | | | | Automation testing can only guarantee a positive customer experience and user- friendliness. | | | | |
| Low return on investment | | | | High return on investment | | | |

**AUTOMATION TESTING METHODOLOGIES**

Automation testing contains the following three different methodologies and approaches, which will help the test engineer to enhance the software product's quality.

* GUI Testing
* Code-Driven
* Test Automation Framework

### GUI (Graphical user interface) Testing

In this approach, we can implement that software or an application, which contains GUIs so that the automation test engineers can record user actions and evaluate them many times. We know that the Test cases can be written in several programming languages like JAVA, C#, Python, Perl, etc.

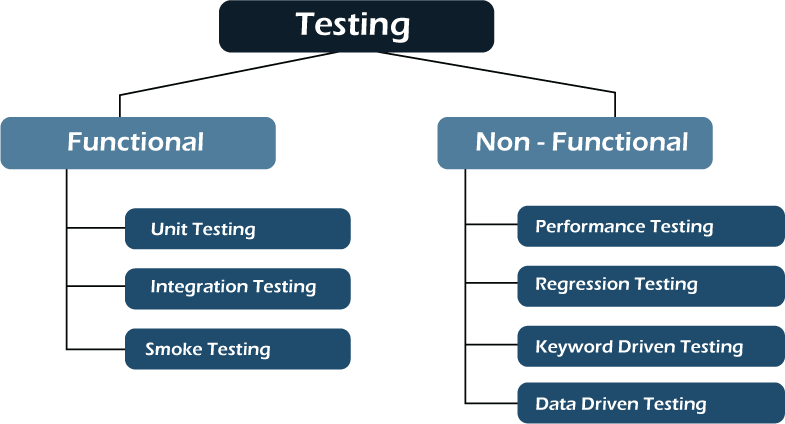
### Code-Driven

The code-driven technique is the subsequent methodology used in automation testing. In this method, the test engineer will mainly concentrate on test case execution in order to identify whether the several parts of code are performing according to the given requirement or not. Hence, it is very a commonly used method in agile software development.

### Test Automation Framework

The test automation framework is a set of rules used to generate valuable results of the automated testing activity. Similarly, it brings together test data sources, function libraries, object details, and other reusable modules.

**FUNCTIONAL AND NON-FUNCTIONAL TESTING**



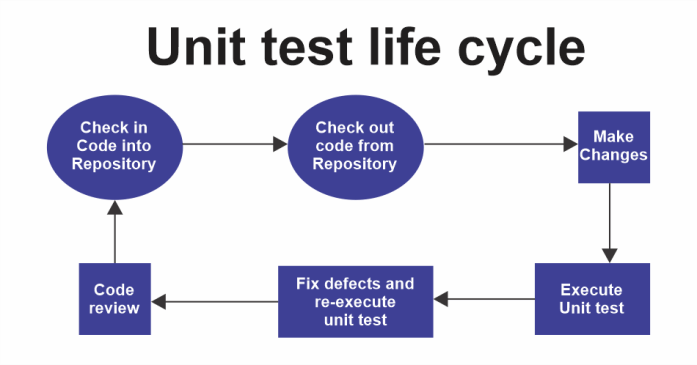
### Functional Testing

The first test performed by tester on newly revised software is called functional testing, which verifies all the software functions' features based on user requirement. This testing works on the real-world business application and obtaining the expected output from a given input. All application functions are tested and it involve smoke, unit, and integration testing.

1. **Unit testing**

The unit is the smallest component of the software that functions individually. Unit testing simplifies the testing of the whole software, where each software element is fully tested before the final version is out. Unit testing depicts how the code performs at each part and has a faster execution time.

It's the favourite of developers because it consumes less time and assure the working of each part of the software. Before automation testing, the developers write the code for testing, but now there is no need. The unit testing technique is divided into three broad categories: White box testing, Black box testing and Grey box testing.

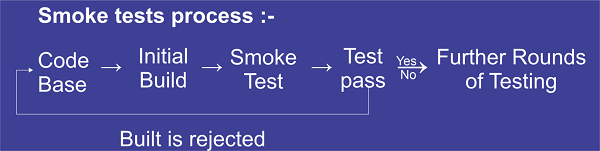


1. **Integration testing**

Integration testing is more complicated to set up compared with other tests. All the modules of the application communicate with each other to perform tasks. Therefore, testers group them for testing and exposing the flaws in maintaining the interaction between these modules. Another name for this testing is I&T or string testing, considered end-to-end.

1. **Smoke testing**

This testing checks and defines the product's stability (whether stable or not). If the product result is unstable, it is called an 'unstable build' and sent back to developers, where they run more test cases to find out the root cause of the problem. The smoke test works like this:-



### NON-FUNCTIONAL TESTING

Non-functional testing focuses on how well application functions are working, not on what the product does. It is the opposite of functional testing, where application elements like reliability, usability, performance, etc., are tested. Some types of non-functional testing are reliability testing, load testing, compatibility testing, performance testing, security testing etc.

1. **Performance testing**

This non-functional testing tests the software's stability, responsiveness and speed under the workload. It finds out the potential issues faced by critical software and medical programs used by the user, like slow operation of software under stressful circumstances. It finds hurdles in the performance of software and removes them to increase the ability of software to deliver the best results to the end user.

1. **Regression testing**

When some changes are made to the code of software or application, it needs to be tested to determine whether the software is working as before the change; for this purpose, testers use automation regression testing to automate scripts, applications of workflows, plans and other activities. It tests the system or software workflow after its updation and functional error.

1. **Keyword-driven testing**

Keyword-driven testing tests the application using the data files consisting of the keywords related to the application, representing a set of actions needed to carry out the step. Here these specific keywords are identified and connected with the specific action. Therefore during testing, when these keywords are used, their related actions will automatically be done. This keyword testing is a popular choice for many businesses as it's flexible, concise, easy to maintain and reusable. Keyword-driven testing is compatible with all kinds of automation tools in the market. Instead of programming experts, functional testers can plan the testing before the application is fully developed.

1. **Data-driven testing**

In data-driven testing, automation is inbuilt and very effective due to the few facilities provided, like the reusability of code, change in the script doesn't affect the test cases, and this testing can be carried out in the phase of the software development cycle. It provides consistency in results and reduces the investment of time and resources. Test cases use the data separately stored in the table or spreadsheet format, and testers have multiple data sets for testing.