UNIVERSITY OF RAJSHAHI

Rajshahi, BANGLADESH.

Course Code:

ICE-3251

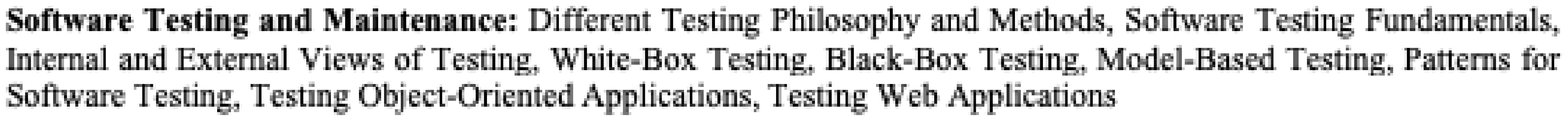
Course Title :

Software Engineering

Software Engineering

ICE-3251

Software Testing and Maintenance



Software Testing and Maintenance: Different Testing Philosophy and Methods, Software Testing Fundamentals, Internal and External Views of Testing, White-Box Testing, Black-Box Testing, Model-Based Testing, Patterns for Software Testing, Testing Object-Oriented Applications, Testing Web Applications

What is Software Testing?

- Once source code has been generated, software must be tested to uncover (and correct) as many errors as possible before delivery to your customer.
- Software Engineer’s goal is to design a series of test cases that have a high likelihood of finding errors.
- Software is tested to uncover errors that were made inadvertently as it was designed and constructed.

How do Engineer conduct the Software Tests?

- Should you develop a formal plan for your tests?
- Should you test the entire program as a whole or run tests only on a small part of it?
- Should you rerun tests you've already conducted as you add new components to a large system?
- When should you involve the customer?

These and many other questions are answered when you develop a software testing strategy.

Who perform Software Testing?

- Usually, a strategy for software testing is developed by the project manager, software engineers, and testing specialists.
- During early stages of testing, a software engineer performs all tests. However, as the testing process progresses, testing specialists may become involved.

Software Testing Strategies

- Strategy for software testing provides a road map that describes the steps to be conducted as part of testing.
- Any testing strategy must incorporate test planning, test-case design, test execution, and resultant data collection and evaluation.
- A software testing strategy should be flexible enough to promote a customized testing approach. At the same time, it must be rigid enough to encourage reasonable planning and management tracking as the project progresses

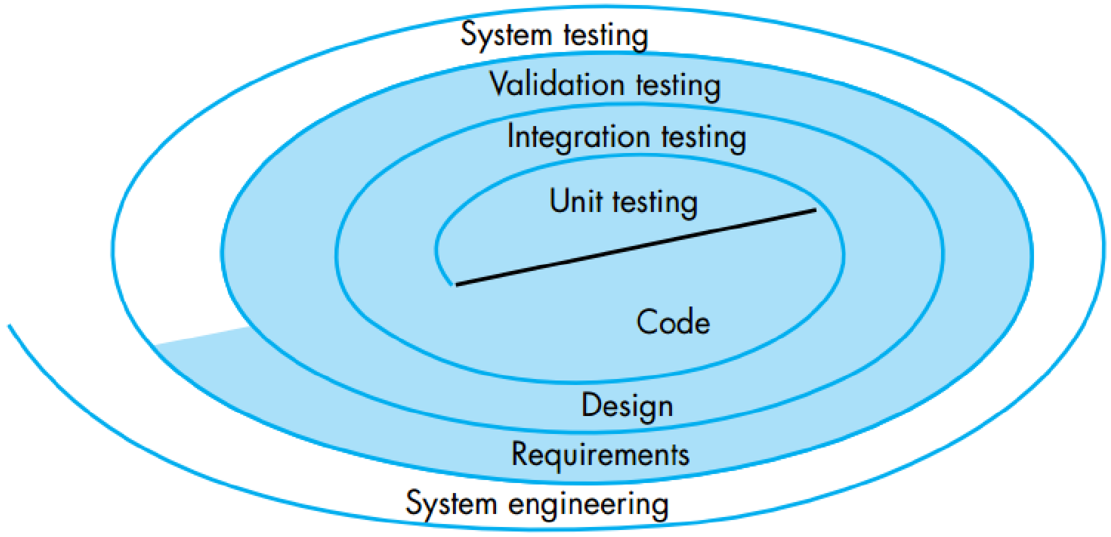
Generic Characteristics of Software Testing Strategies

- To perform effective testing, you should conduct effective technical reviews. By doing this, many errors will be eliminated before testing commences.
- Testing begins at the component level and works "outward" toward the integration of the entire computer-based system.
- Different testing techniques are appropriate for different software engineering approaches and at different points in time.
- Testing is conducted by the developer of the software and (for large projects) an independent test group.
- Testing and debugging are different activities, but debugging must be accommodated in any testing strategy.

Verification and Validation

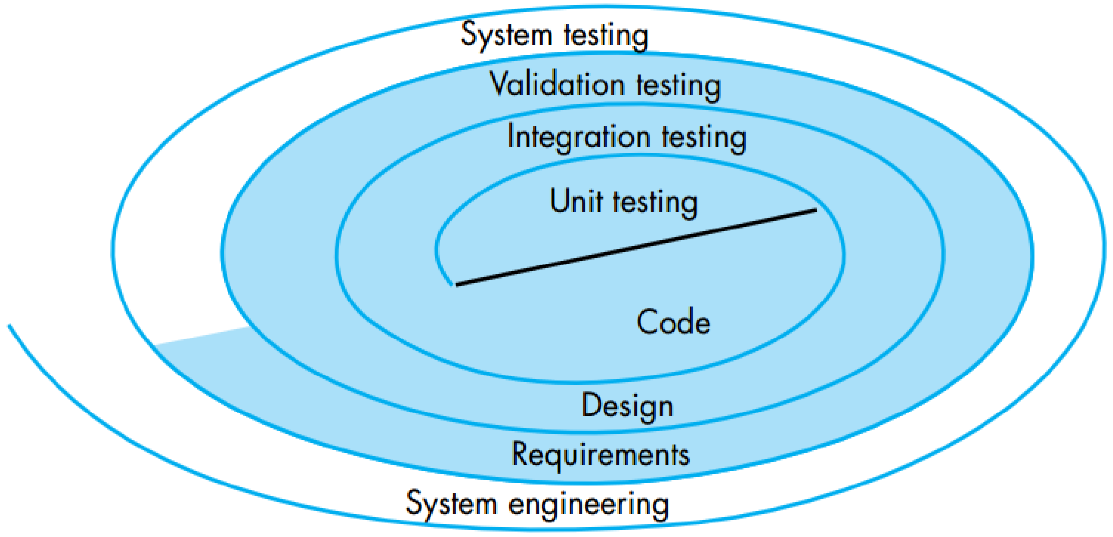
- Software testing is one element of a broader topic that is often referred to as verification and validation (V&V).
- **Verification** refers to the set of tasks that ensure that software correctly implements a specific function.
- **Validation** refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.
- **Boehm states this another way:**
    - Verification: “Are we building the product right?”
    - Validation: “Are we building the right product?”
- The definition of V&V encompasses many software quality assurance activities

Software Testing Process



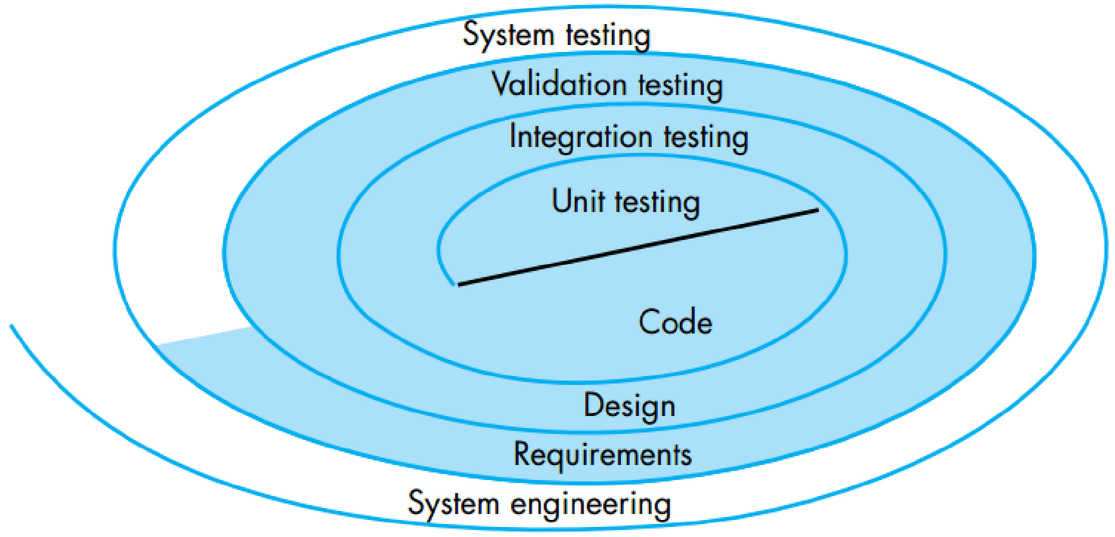
- The software process may be viewed as the spiral illustrated in the above Figure.
- Initially, system engineering defines the role of software and leads to software requirements analysis, where the information domain, function, behavior, performance, constraints, and validation criteria for software are established.

Software Testing Process



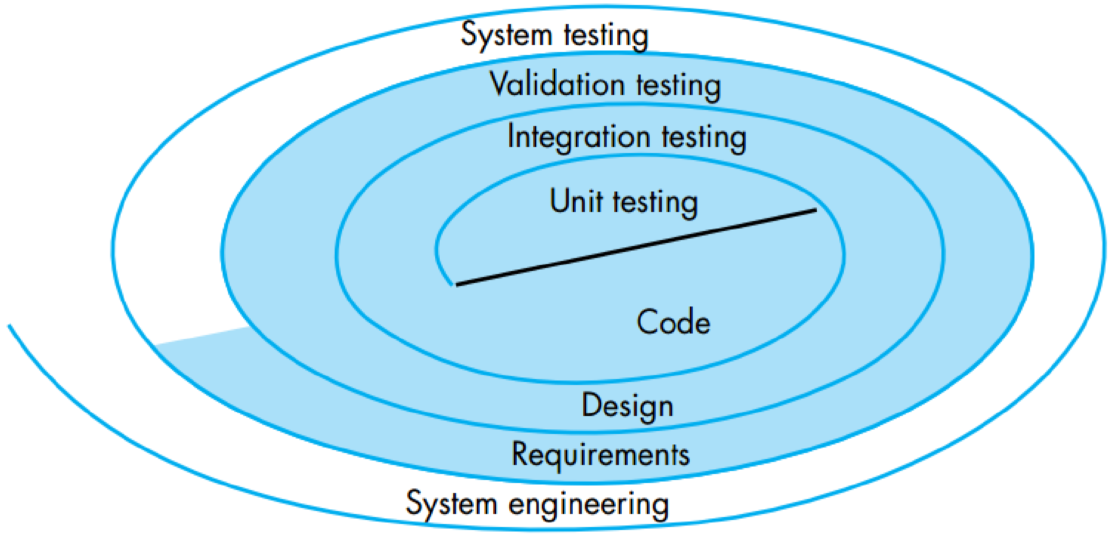
- Moving inward along the spiral, you come to design and finally to coding.
- To develop computer software, you spiral inward along streamlines that decrease the level of abstraction on each turn.

Strategies for Software Testing



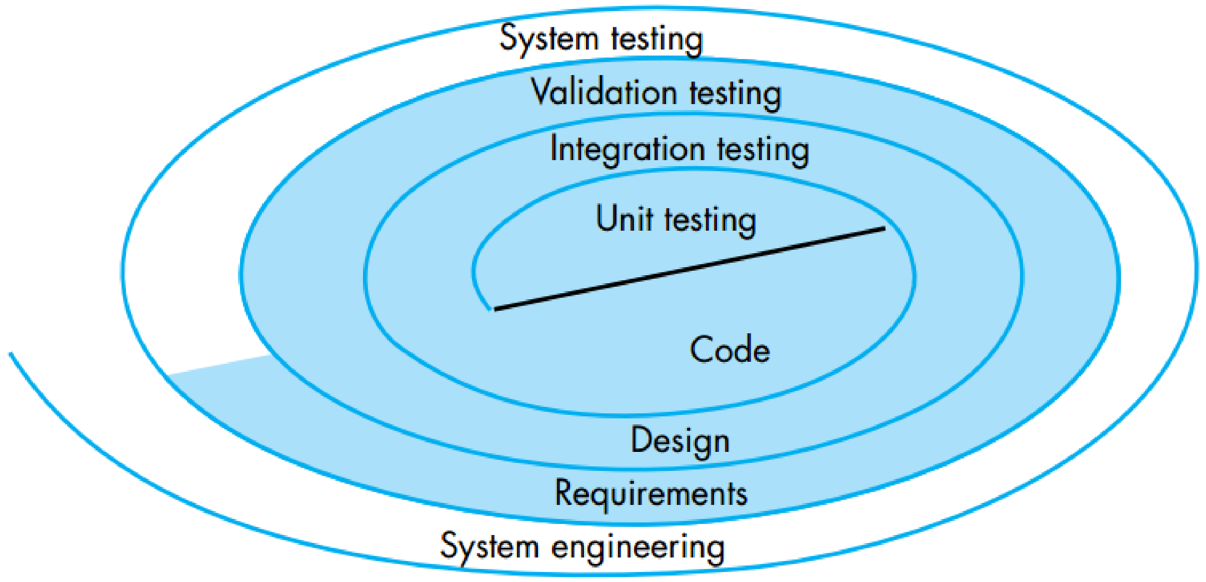
- A strategy for software testing may also be viewed in the context of the spiral (above figure).
- Unit testing begins at the vortex of the spiral and concentrates on each unit (e.g., component, class, or WebApp content object) of the software as implemented in source code.

Strategies for Software Testing



- Testing progresses by moving outward along the spiral to integration testing, where the focus is on design and the construction of the software architecture.
- Taking another turn outward on the spiral, you encounter validation testing, where requirements established as part of requirements modeling are validated against the software that has been constructed.

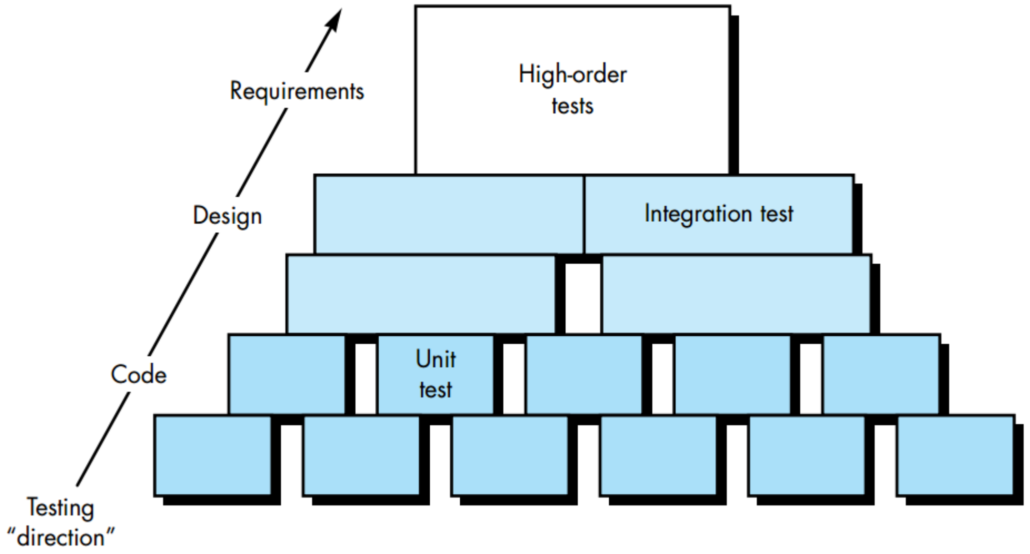
Strategies for Software Testing



- Finally, you arrive at system testing, where the software and other system elements are tested as a whole.
- To test computer software, you spiral out along streamlines that broaden the scope of testing with each turn.

Software Testing Steps

- Actually, Software testing incorporates a series of four steps that are implemented sequentially.

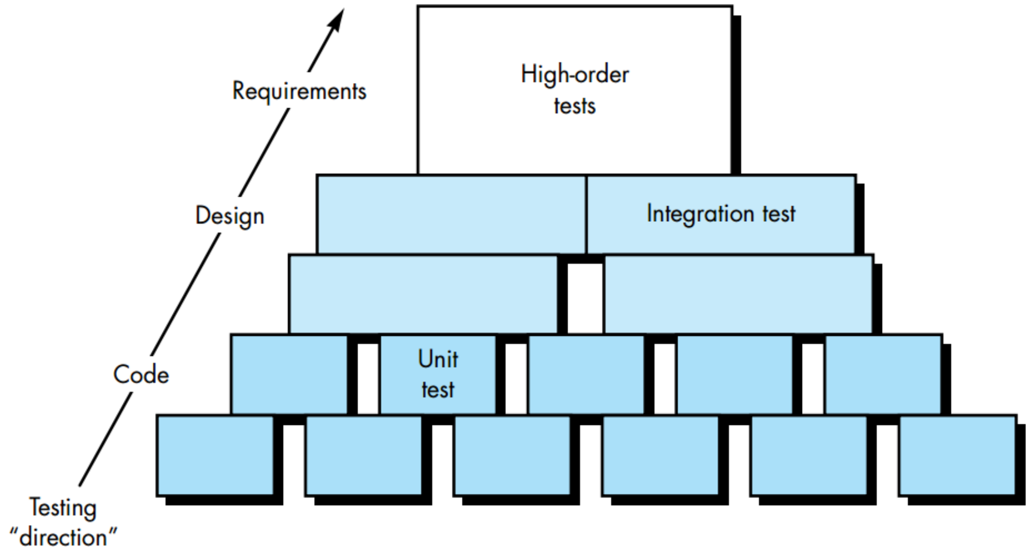


Unit Testing:

- Initially, tests focus on each component individually, ensuring that it functions properly as a unit. Hence, the name unit testing.
- Unit testing makes heavy use of testing techniques that exercise specific paths in a component's control structure to ensure complete coverage and maximum error detection.

Software Testing Steps

- Actually, Software testing incorporates a series of four steps that are implemented sequentially.

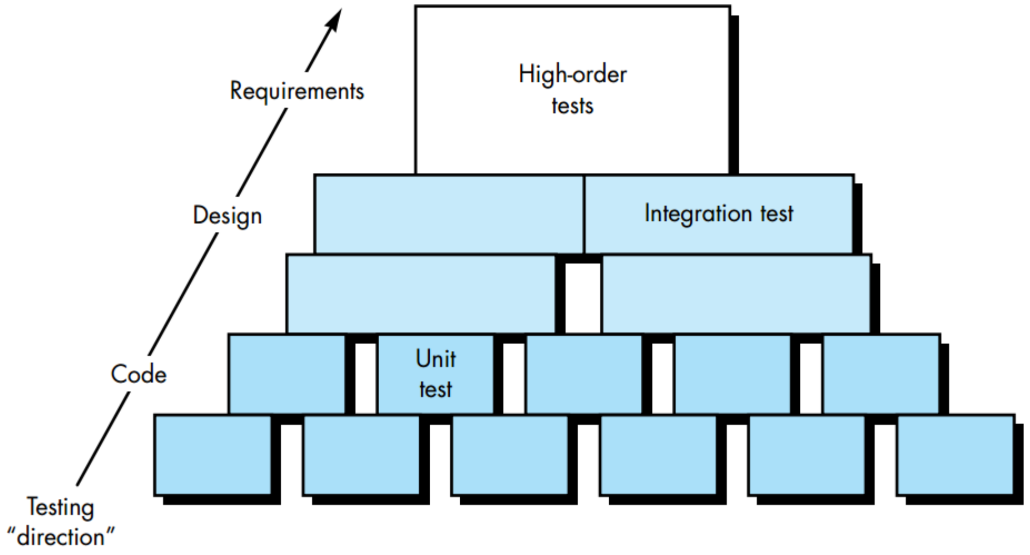


Integration Testing:

- Next, components must be assembled or integrated to form the complete software package.
- Integration testing addresses the issues associated with the dual problems of verification and program construction.

Software Testing Steps

- Actually, Software testing incorporates a series of four steps that are implemented sequentially.

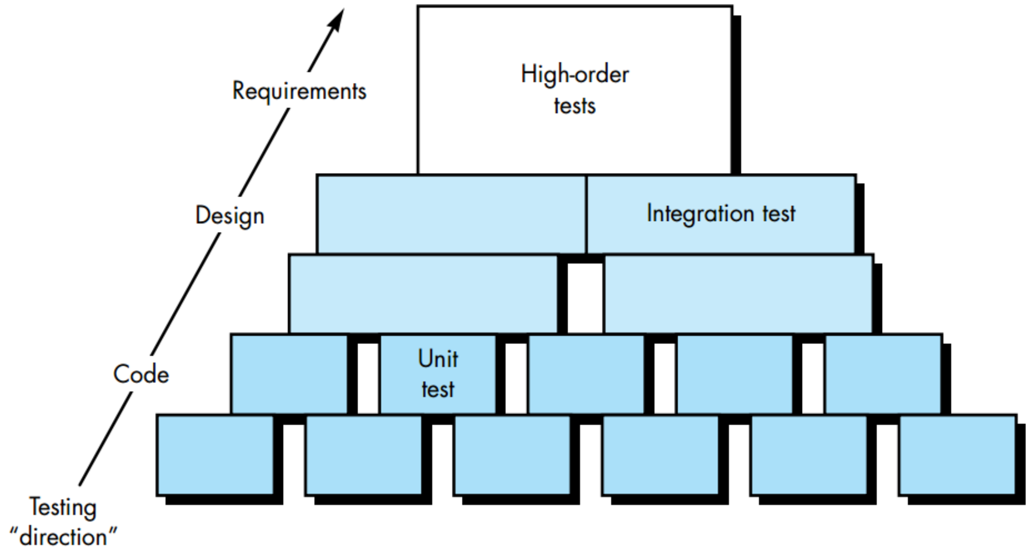


High-Order Testing:

- After the software has been integrated (constructed), a set of high-order tests is conducted.

Software Testing Steps

- Actually, Software testing incorporates a series of four steps that are implemented sequentially.



Validation Testing:

- Validation criteria (established during requirements analysis) must be evaluated.
- Validation testing provides final assurance that software meets all functional, behavioral, and performance requirements.

Goal of Software Testing?

- The goal of testing is to find errors, and
- A good test is one that has a high probability of finding an error.
- Therefore, you should design and implement a computer-based system or a product with “testability*” in mind.
- At the same time, the tests themselves must exhibit a set of characteristics that achieve the goal of finding the most errors with a minimum of effort.

*Software testability is simply how easily a computer program can be tested.

Characteristics of a Testable Software

- **Operability.** “The better it works, the more efficiently it can be tested.”, If a system is designed and implemented with quality in mind, relatively few bugs will block the execution of tests, allowing testing to progress without fits and starts.

- **Observability**. “What you see is what you test.” Inputs provided as part of testing produce distinct outputs. System states and variables are visible or quarriable during execution. Incorrect output is easily identified. Internal errors are automatically detected and reported. Source code is accessible.

Characteristics of a Testable Software

- **Controllability.** "The better we can control the software, the more the testing can be automated and optimized." All possible outputs can be generated through some combination of input, and I/O formats are consistent and structured. All code is executable through some combination of input. Software and hardware states and variables can be controlled directly by the test engineer. Tests can be conveniently specified, automated, and reproduced.
- **Decomposability.** "By controlling the scope of testing, we can more quickly isolate problems and perform smarter retesting." The software system is built from independent modules that can be tested independently.

Characteristics of a Testable Software

Simplicity. “The less there is to test, the more quickly we can test it.” The program should exhibit functional simplicity (e.g., the feature set is the minimum necessary to meet requirements); structural simplicity (e.g., architecture is modularized to limit the propagation of faults), and code simplicity (e.g., a coding standard is adopted for ease of inspection and maintenance).

Characteristics of a Testable Software

Stability. “The fewer the changes, the fewer the disruptions to testing.” Changes to the software are infrequent, controlled when they do occur, and do not invalidate existing tests. The software recovers well from failures.

Understandability. “The more information we have, the smarter we will test.” The architectural design and the dependencies between internal, external, and shared components are well understood. Technical documentation is instantly accessible, well organized, specific and detailed, and accurate. Changes to the design are communicated to testers.

Attributes of a Good Software Test

- ***A good test has a high probability of finding an error.*** To achieve this goal, the tester must understand the software and attempt to develop a mental picture of how the software might fail.
- ***A good test is not redundant.*** Testing time and resources are limited. There is no point in conducting a test that has the same purpose as another test. Every test should have a different purpose.

Attributes of a Good Software Test

- ***A good test should be "best of breed"*** . In a group of tests that have a similar intent, time and resource limitations may dictate the execution of only those tests that has the highest likelihood of uncovering a whole class of errors.
- ***A good test should be neither too simple nor too complex***. Although it is sometimes possible to combine a series of tests into one test case, the possible side effects associated with this approach may mask errors. In general, each test should be executed separately.

Internal Vs. External Views of Testing

Any engineered product (and most other things) can be tested in one of two ways:

(1) (**Black-Box Testing**) Knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational while at the same time searching for errors in each function. This approach takes an external view and is called **black-box testing**.

(2) (**White-Box Testing**) Knowing the internal workings of a product, tests can be conducted to ensure that internal operations are performed according to specifications and all internal components have been adequately exercised. This approach takes an internal view and is termed **white-box testing**.

Black-box testing vs. White-box testing

- Black-box testing tests the matters that are conducted at the software interface. A black-box test examines some fundamental aspect of a system with little regard for the internal logical structure of the software. On the other hand,
- White-box testing of software is predicated on close examination of procedural detail. Logical paths through the software and collaborations between components are tested by exercising specific sets of conditions and/or loops. White-box testing, sometimes called glass-box testing or structural testing,

Model-Based Testing

- Model-based testing (MBT) is a black-box testing technique that uses information contained in the requirements model as the basis for the generation of test cases.
- The MBT technique requires five steps:
  - ➢ Analyze an existing behavioral model for the software or create one.
  - ➢ Traverse the behavioral model and specify the inputs that will force the software to make the transition from state to state.
  - ➢ Review the behavioral model and note the expected outputs as the software makes the transition from state to state.
  - ➢ Execute the test cases.
  - ➢ Compare actual and expected results and take corrective action as required

Patterns for Software Testing?

- Testing patterns describe common testing problems and solutions that can assist you in dealing with them.
- The following three testing patterns (presented in abstract form only) provide representative examples:

✓Pattern names: **PairTesting, SeperateTestInterface,** and **Scenario Testing**

✓Pattern name: **PairTesting**

A process-oriented pattern, pair testing describes a technique that is analogous to pair programming in which two testers work together to design and execute a series of tests that can be applied to unit, integration or validation testing activities.

Patterns for Software Testing?

- ✓ Pattern name: **SeparateTestInterface**

  There is a need to test every class in an object-oriented system, including "internal classes" (i.e., classes that do not expose any interface outside of the component that used them). The SeparateTestInterface pattern describes how to create "a test interface that can be used to describe specific tests on classes that are visible only internally to a component"

- ✓ Pattern name: **ScenarioTesting**

  Once unit and integration tests have been conducted, there is a need to determine whether the software will perform in a manner that satisfies users. The ScenarioTesting pattern describes a technique for exercising the software from the user's point of view. A failure at this level indicates that the software has failed to meet a user visible requirement

Testing Object Oriented System

- Testing is a continuous activity during software development.
- In object-oriented systems, testing encompasses three levels, namely:,
    - *unit testing*,
    - *subsystem testing*, and
    - *system testing*.

Unit Testing

In unit testing, the individual classes are tested. It is seen whether the class attributes are implemented as per design and whether the methods and the interfaces are error-free. Unit testing is the responsibility of the application engineer who implements the structure.

Testing Object Oriented System

- In object-oriented systems, testing encompasses three levels, namely, *unit testing*, *subsystem testing*, and *system testing*.

Subsystem Testing

This involves testing a particular module or a subsystem and is the responsibility of the subsystem lead. It involves testing the associations within the subsystem as well as the interaction of the subsystem with the outside. Subsystem tests can be used as regression tests for each newly released version of the subsystem.

System Testing

System testing involves testing the system as a whole and is the responsibility of the quality-assurance team. The team often uses system tests as regression tests when assembling new releases.

What is Web Application Testing?

- Web Application testing, or website testing checks owner's web application or website for potential bugs before its made live and is accessible to general public.
- Web Testing checks for functionality, usability, security, compatibility, performance of the web application or website.

What is web application testing ?

- In Software Engineering, the following testing types/technique may be performed depending on your web testing requirements.
- There are 8 steps guide to website or web application testing:
  - **Functionality Testing of a Website**
  - **Usability testing**
  - **Interface Testing**
  - **Compatibility testing**
  - **Performance Testing**
  - **Security testing**
  - **Crowd Testing**

Steps Guides to Web Application Testing

1. **Functionality Testing of a Website**

- It is a process that includes several testing parameters like user interface, APIs, database testing, security testing, client and server testing and basic website functionalities. Functional testing is very convenient and it allows users to perform both manual and automated testing. It is performed to test the functionalities of each feature on the website.
- **Example of Functional Test Scenarios:** Test the functionality of the buttons available, Test the timeout functionality, Test the Java script is properly working in different browsers (IE, Firefox, Chrome, safari and Opera), Test to see what happens if a user deletes cookies while in the site, Test to see what happens if a user deletes cookies after visiting a site, Test all the data inside combo/list box is arranged in chronological order.

Steps Guides to Web Application Testing

2. Usability testing of a Website

Usability Testing has now become a vital part of any web based project. It can be **carried out by testers** like you **or a small focus group** similar to the target audience of the web application.

Example of Usability testing Scenarios: Web page content should be correct without any spelling or grammatical errors, All fonts should be same as per the requirements, All the text should be properly aligned, All the error messages should be correct without any spelling or grammatical errors and the error message should match with the field label, Tool tip text should be there for every field, All the fields should be properly aligned.

Steps Guides to Web Application Testing

3. Interface Testing of a website:

- Three areas to be tested here are - Application, Web and Database Server.
  - **Application:** Test requests are sent correctly to the Database and output at the client side is displayed correctly. Errors if any must be caught by the application and must be only shown to the administrator and not the end user.
  - **Web Server**: Test Web server is handling all application requests without any service denial.
  - **Database Server:** Make sure queries sent to the database give expected results.

Steps Guides to Web Application Testing

4. Database Testing of a website:

Database is one critical component of your web application and stress must be laid to test it thoroughly. Testing activities will include-

- Test if any errors are shown while executing queries
- Data Integrity is maintained while creating, updating or deleting data in database.
- Check response time of queries and fine tune them if necessary.
- Test data retrieved from your database is shown accurately in your web application

Example Test Cases for Database Testing: Verify the database name: The database name should match with the specifications, Verify the Tables, columns, column types and defaults: All things should match with the specifications, Verify whether the column allows a null or not, Verify the Primary and foreign key of each table, Verify the Stored Procedure: Test whether the Stored procedure is installed or not, Verify the Stored procedure name

Steps Guides to Web Application Testing

5.Compatibility Testing of a website:

- Compatibility tests ensures that your web application displays correctly across different devices. This would include-
- **Browser Compatibility Test**: Same website in different browsers will display differently. You need to test if your web application is being displayed correctly across browsers, JavaScript, AJAX and authentication is working fine. You may also check for Mobile Browser Compatibility. Make sure your website works fine for various combination of Operating systems such as Windows, Linux, Mac and Browsers such as Firefox, Internet Explorer, Safari etc.

•**Example Test Cases for Compatibility Testing:** Test the website in different browsers (IE, Firefox, Chrome, Safari and Opera) and ensure the website is displaying properly, Test the HTML version being used is compatible with appropriate browser versions, Test the images display correctly in different browsers, Test the fonts are usable in different browsers, Test the java script code is usable in different browsers, Test the Animated GIF's across different browsers.

Steps Guides to Web Application Testing

6. Performance Testing of a website:

- This will ensure your site works under all loads. Software Testing activities will include but not limited to –
  - Website application response times at different connection speeds
  - Load test your web application to determine its behavior under normal and peak loads
  - Stress test your web site to determine its break point when pushed to beyond normal loads at peak time.
  - Test if a crash occurs due to peak load, how does the site recover from such an event
  - Make sure optimization techniques like gzip compression, browser and server side cache enabled to reduce load times

Steps Guides to Web Application Testing

7. Security Testing of a website:

Security Testing is vital for e-commerce website that store sensitive customer information like credit cards. Testing Activities will include-

- Test unauthorized access to secure pages should not be permitted
- Restricted files should not be downloadable without appropriate access
- Check sessions are automatically killed after prolonged user inactivity
- On use of SSL certificates, website should re-direct to encrypted SSL pages.

Sample Test Scenarios for Security Testing:

- Verify the web page which contains important data like password, credit card numbers, secret answers for security question etc should be submitted via HTTPS (SSL).
- Verify the important information like password, credit card numbers etc should display in encrypted format.
- Verify if the password is changed the user should not be able to login with the old password.
- Verify if the user is logged out from the system or user session was expired, the user should not be able to navigate the site.

Test Strategies for WebApps

- The strategy for WebApp testing adopts the basic principles for all software testing and applies a strategy and tactics that are used for object-oriented systems.
- The following steps summarize the approach:
  - The content model for the WebApp is reviewed to uncover errors.
  - The interface model is reviewed to ensure that all use cases can be accommodated.
  - The design model for the WebApp is reviewed to uncover navigation errors.
  - The user interface is tested to uncover errors in presentation and/or navigation mechanics.
  - Each functional component is unit tested.
  - Navigation throughout the architecture is tested.
  - The WebApp is implemented in a variety of different environmental configurations and is tested for compatibility with each configuration.
  - Security tests are conducted in an attempt to exploit vulnerabilities in the WebApp or within its environment

Test Strategies for MobileApps

The strategy for testing mobile applications adopts the basic principles for all software testing.

•***User-experience testing.*** Users are involved early in the development process to ensure that the MobileApp lives up to the usability and accessibility expectations of the stakeholders on all supported devices.

•***Device compatibility testing***. Testers verify that the MobileApp works correctly on all required hardware and software combinations.

•***Performance testing***. Testers check nonfunctional requirements unique to mobile devices (e.g., download times, processor speed, storage capacity, power availability).

•***Connectivity testing.*** Testers ensure that the MobileApp can access any needed networks or Web services and can tolerate weak or interrupted network access.

•***Security testing***. Testers ensure that the MobileApp does not compromise the privacy or security requirements of its users.

•***Testing-in-the-wild***. The app is tested under realistic conditions on actual user devices in a variety of networking environments around the globe.

•***Certification testing***. Testers ensure that the MobileApp meets the standards established by the app stores that will distribute it.