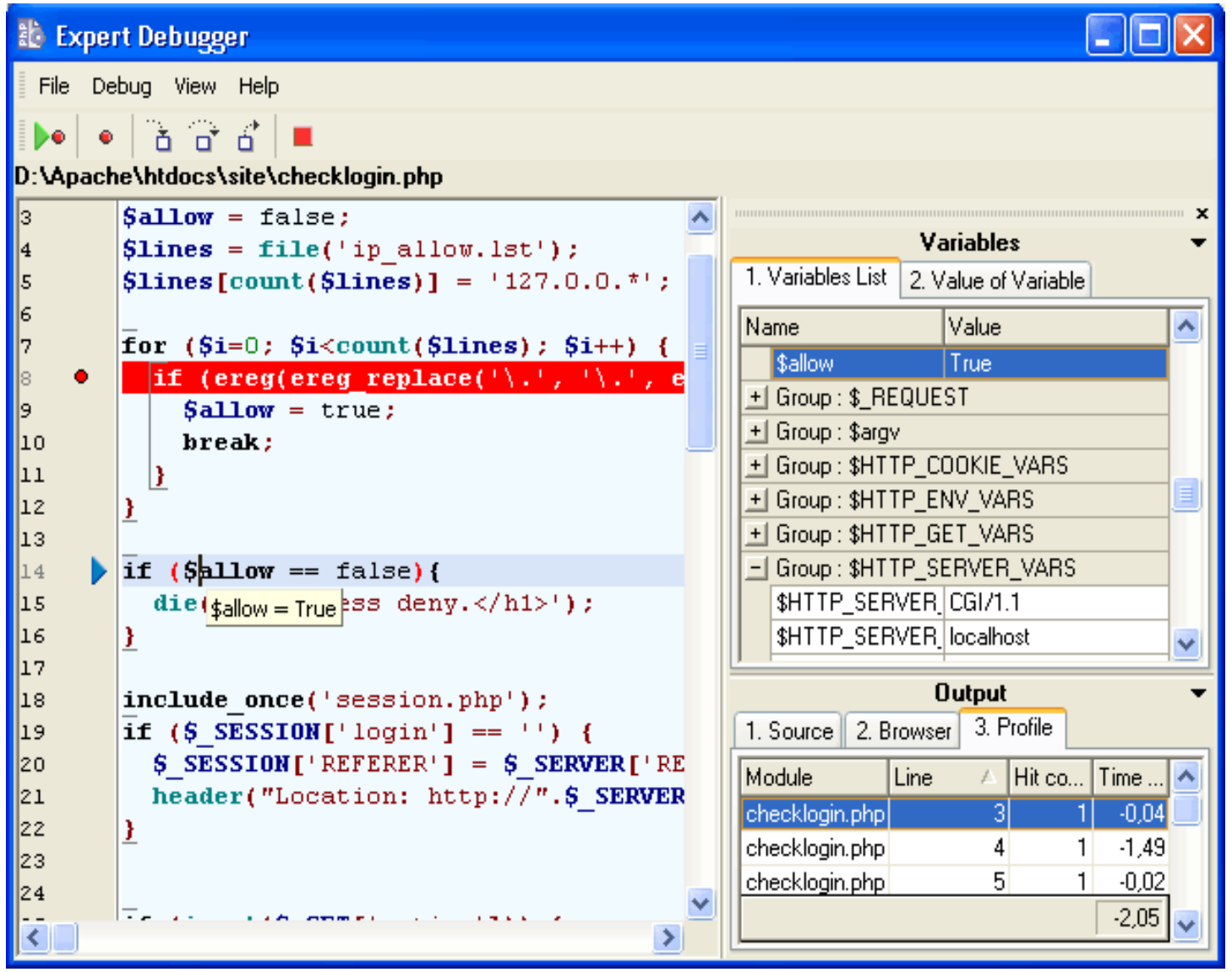
Debugging, Integration and System Testing

What is debugging

- Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system



Need for debugging

Once errors are identified in a program code, it is necessary to first identify the precise program statements responsible for the errors and then to fix them. Identifying errors in a program code and then fix them up are known as debugging.

Debugging approaches

The following are some of the approaches popularly adopted by programmers for debugging.

Brute Force Method:

This is the most common method of debugging but is the least efficient method. In this approach, the program is loaded with print statements to print the intermediate values with the hope that some of the printed values will help to identify the statement in error. This approach becomes more systematic with the use of a symbolic debugger (also called a source code debugger), because values of different variables can be easily checked and break points and watch points can be easily set to test the values of variables effortlessly.

Backtracking:

This is also a fairly common approach. In this approach, beginning from the statement at which an error symptom has been observed, the source code is traced backwards until the error is discovered. Unfortunately, as the number of source lines to be traced back increases, the number of potential backward paths increases and may become unmanageably large thus limiting the use of this approach.

Cause Elimination Method:

In this approach, a list of causes which could possibly have contributed to the error symptom is developed and tests are conducted to eliminate each. A related technique of identification of the error from the error symptom is the software fault tree analysis.

Program Slicing:

This technique is similar to back tracking. Here the search space is reduced by defining slices. A slice of a program for a particular variable at a particular statement is the set of source lines preceding this statement that can influence the value of that variable [Mund2002].

Debugging guidelines

Debugging is often carried out by programmers based on their ingenuity. The following are some general guidelines for effective debugging:

- Many times debugging requires a thorough understanding of the program design. Trying to debug based on a partial understanding of the system design and implementation may require an inordinate amount of effort to be put into debugging even simple problems.
- Debugging may sometimes even require full redesign of the system. In such cases, a common mistakes that novice programmers often make is attempting not to fix the error but its symptoms.
- One must be beware of the possibility that an error correction may introduce new errors. Therefore after every round of error-fixing, regression testing must be carried out.

Testing Levels (Different Types)

- **Unit Testing**
- **Integression Testing**
- **System Testing**
- **Regression Testing**

Unit Testing

- During this first round of testing, the program is submitted to assessments that focus on specific units or components of the software to determine whether each one is fully functional.
- The main aim of this endeavor is to **determine whether the application functions as designed**.
- In this phase, a unit can refer to a function, individual program or even a procedure, and a White-box Testing method is usually used to get the job done.
- One of the biggest benefits of this testing phase is that it can be run every time a piece of code is changed, allowing issues to be resolved as quickly as possible.

Integration Testing

- Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group.
- **This testing level is designed to find interface defects between the modules/functions**.
- This is particularly beneficial because it determines how efficiently the units are running together.
- Keep in mind that no matter how efficiently each unit is running, if they aren't properly integrated, it will affect the functionality of the software program.

Integration test approaches

There are four types of integration testing approaches. Any one (
the following approaches can be used to develop the integration t
approaches are the following:

- Big bang approach
- Top-down approach
- Bottom-up approach
- Mixed-approach

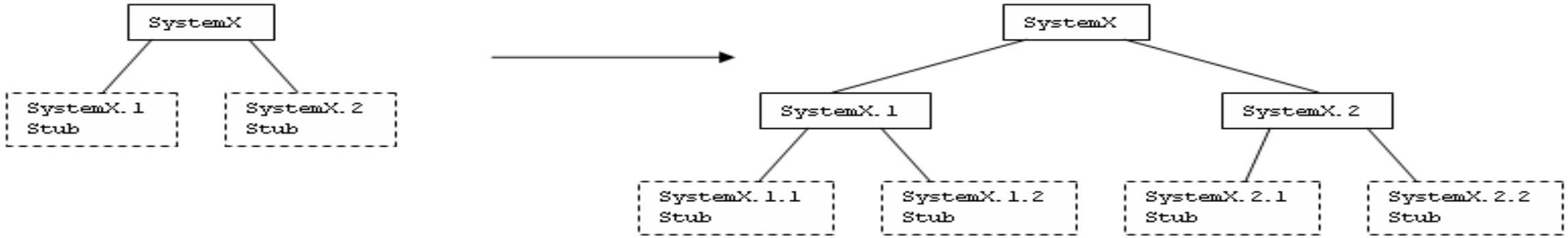
Big Bang and Top Down

- Big Bang is an approach to Integration Testing where all or most f the units are combined together and tested at one go. This approach is taken when the testing team receives the entire software in a bundle. So what is the difference between Big Bang Integration Testing and System Testing? Well, the former tests only the interactions between the units while the latter tests the entire system.
- Top Down is an approach to Integration Testing where top-level units are tested first and lower level units are tested step by step after that. This approach is taken when top-down development approach is followed. Test Stubs are needed to simulate lower level units which may not be available during the initial phases.

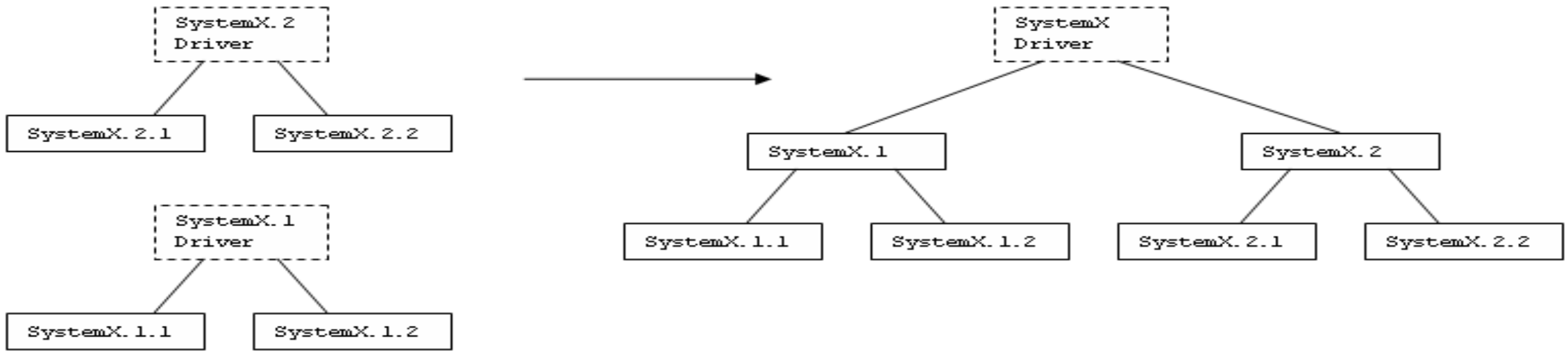
Bottom up and Hybrid

- **Bottom Up** is an approach to Integration Testing where bottom level units are tested first and upper-level units step by step after that. This approach is taken when bottom-up development approach is followed. Test Drivers are needed to simulate higher level units which may not be available during the initial phases.
- **Sandwich/Hybrid** is an approach to Integration Testing which is a combination of Top Down and Bottom Up approaches.

Top-Down Integration Testing

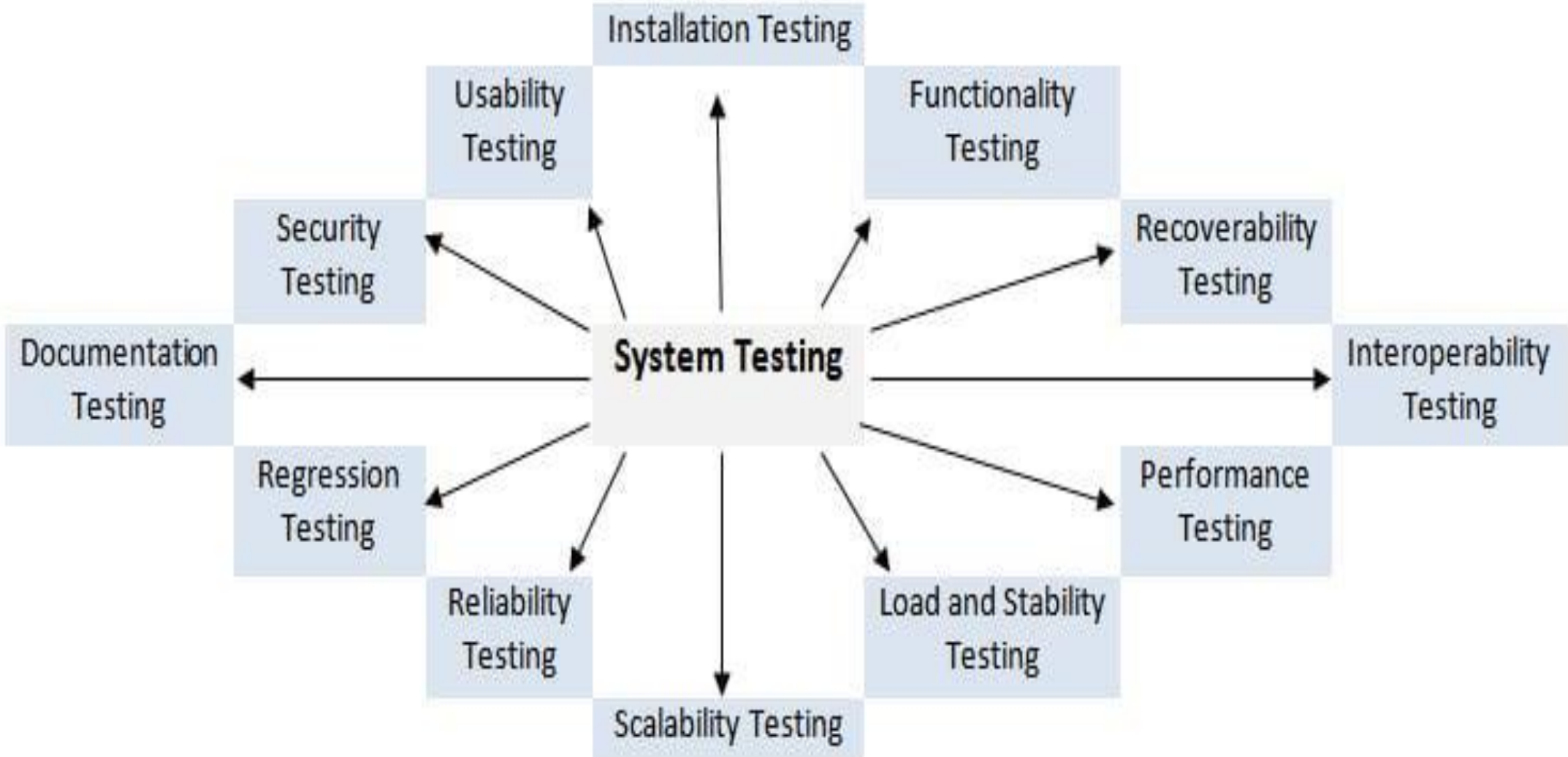


Bottom-Up Integration Testing



System Testing

- System testing is the first level in which the complete application is tested as a whole.
- **The goal at this level is to evaluate whether the system has complied with all of the outlined requirements** and to see that it meets Quality Standards.
- System testing is undertaken by independent testers who haven't played a role in developing the program.
- This testing is performed in an environment that **closely mirrors production**.
- System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the customer.


Installation Testing
Usability Testing
Functionality Testing
Security Testing
Recoverability Testing
Documentation Testing
System Testing
Interoperability Testing
Regression Testing
Performance Testing
Reliability Testing
Load and Stability Testing
Scalability Testing

System testing

System tests are designed to validate a fully developed system to assure that it meets its requirements. There are essentially three main kinds of system testing:

- **Alpha Testing.** Alpha testing refers to the system testing carried out by the test team within the developing organization.

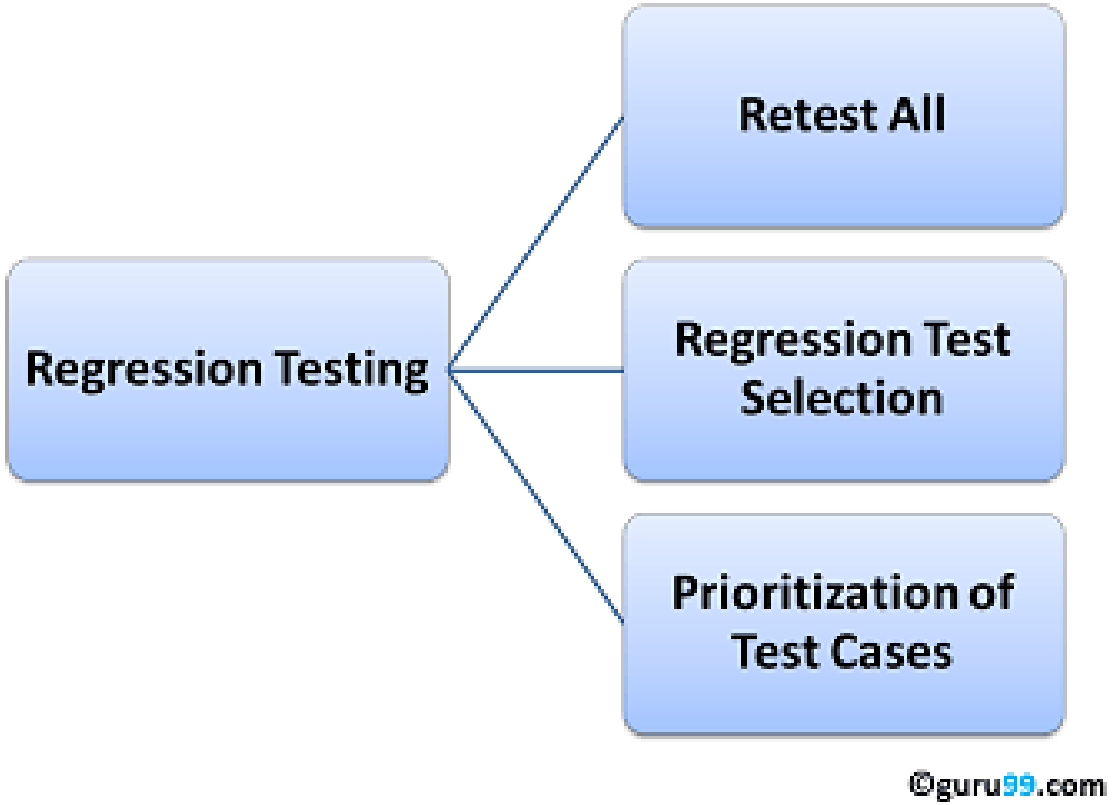
- **Beta testing.** Beta testing is the system testing performed by a select group of friendly customers.

- **Acceptance Testing.** Acceptance testing is the system testing performed by the customer to determine whether he should accept the delivery of the system.

In each of the above types of tests, various kinds of test cases are designed by referring to the SRS document. Broadly, these tests can be classified into functionality and performance tests. The functionality tests test the functionality of the software to check whether it satisfies the functional requirements as documented in the SRS document. The performance tests test the conformance of the system with the nonfunctional requirements of the system.

Regression testing

- Regression testing is the process of testing changes to computer programs to **make sure that the older programming still works with the new changes.**
- Regression testing is a normal part of the program development process and, in larger companies, is done by code testing specialists.

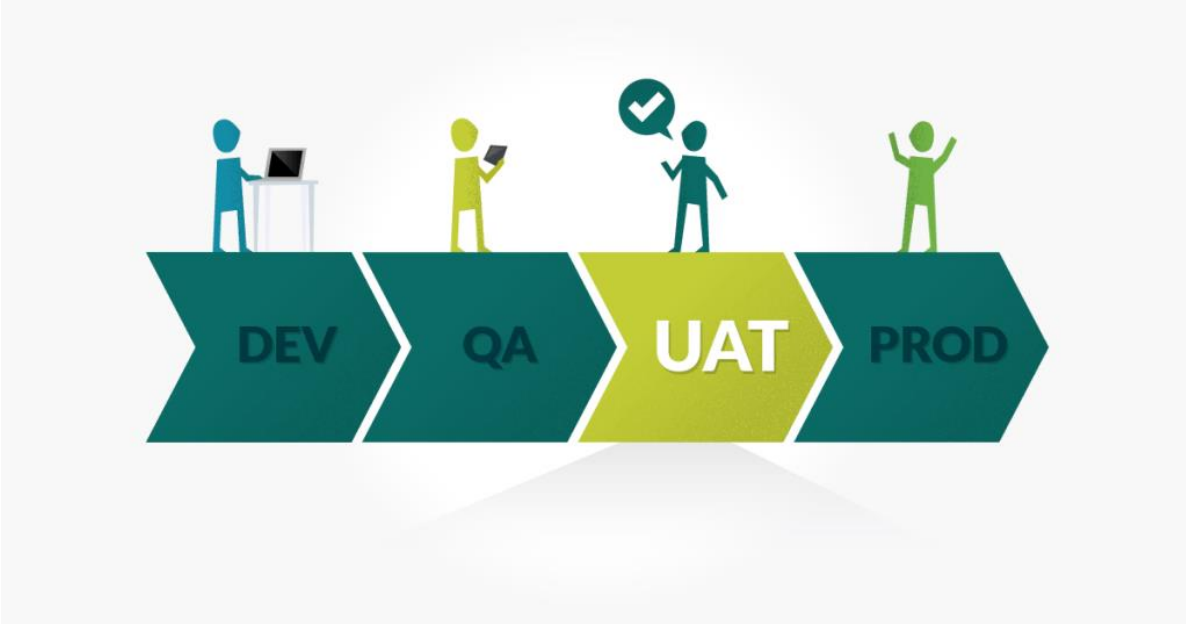


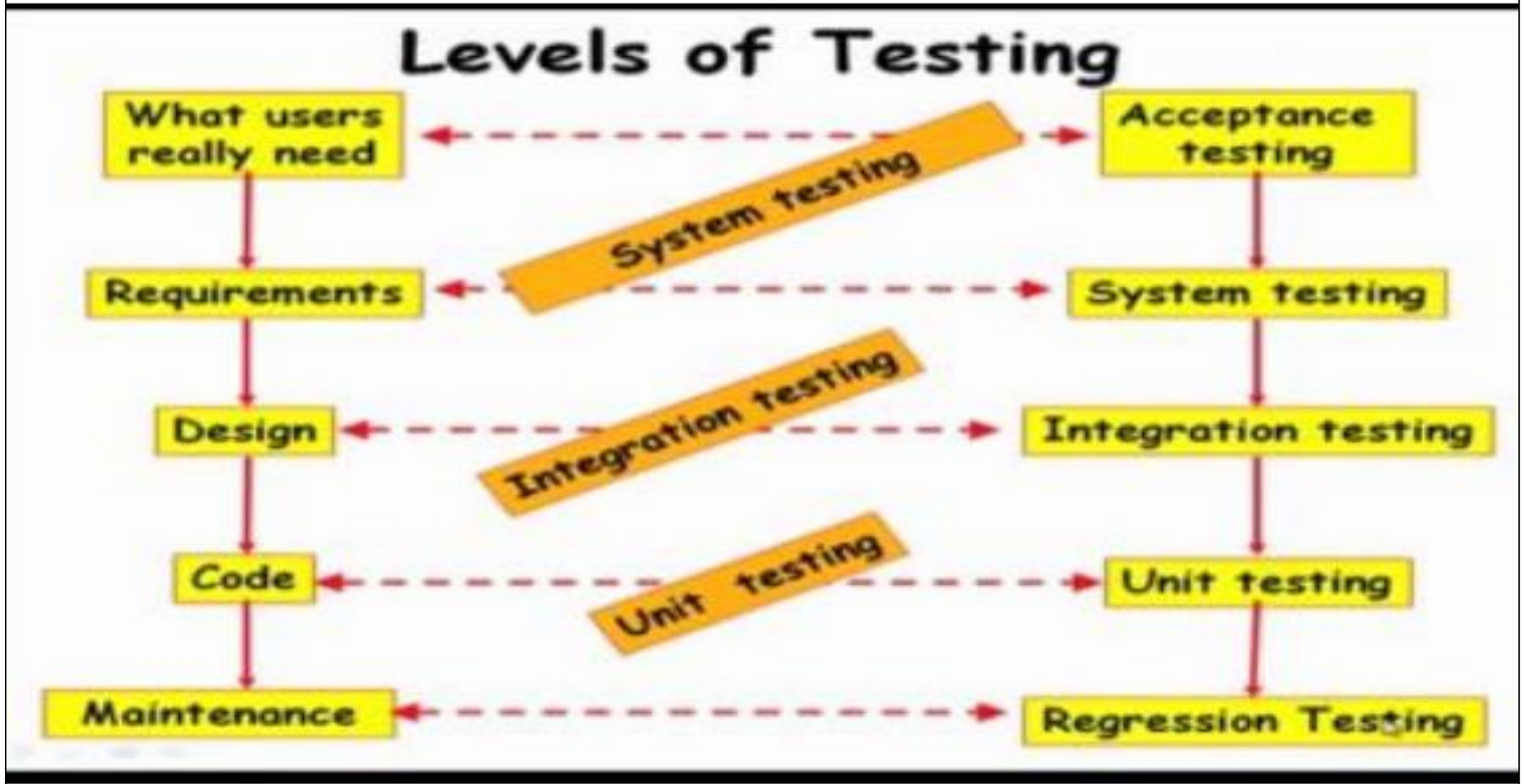
Test Levels

- Unit Testing - done by developers
- Integration Testing - done by QA team
- System testing - done by QA team
- Acceptance Testing - Done by customers

ACCEPTANCE TESTING

- ACCEPTANCE TESTING is a level of software testing where a system is tested for acceptability.
- The purpose of this test is to **evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.**
- Done by customers




Levels of Testing
What users really need
Acceptance testing
System testing
Requirements
System testing
Integration testing
Design
Integration testing
Code
Unit testing
Unit testing
Maintenance
Regression Testing

Overview of Activities During System and Integration Testing

- Test Suite Design
- Run test cases
- Check results to detect failures.
- Prepare failure list

Tester

- Debug to locate errors
- Correct errors.

Developer

Examples

- Unit test - error in algorithm.
- Integration testing - Mismatch of parameters, their order, number of data types.
- System testing - Performance bugs, usability issues.

Performance testing

Performance testing is carried out to check whether the system needs the non-functional requirements identified in the SRS document. There are several types of performance testing. Among of them nine types are discussed below. The types of performance testing to be carried out on a system depend on the different non-functional requirements of the system documented in the SRS document. All performance tests can be considered as black-box tests.

- Stress testing
- Volume testing
- Configuration testing
- Compatibility testing
- Regression testing
- Recovery testing
- Maintenance testing
- Documentation testing
- Usability testing