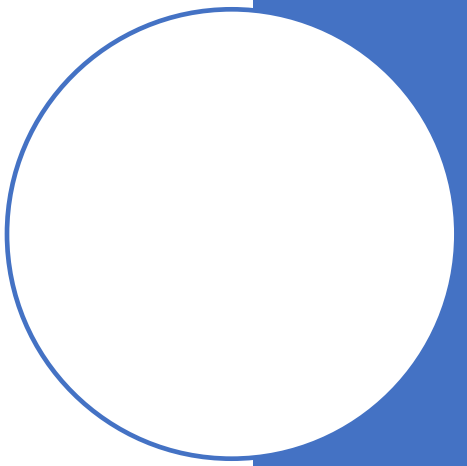
Test Case Generation & Checking Effectiveness

Can be Domain specific, Random, Source-code based to test execution paths in control flow graph, and requirement specification based.

Software Testing Metric: Code coverage, branch coverage, path coverage etc.

Measuring Test cases Effectiveness: Mutation Testing



Software Testing

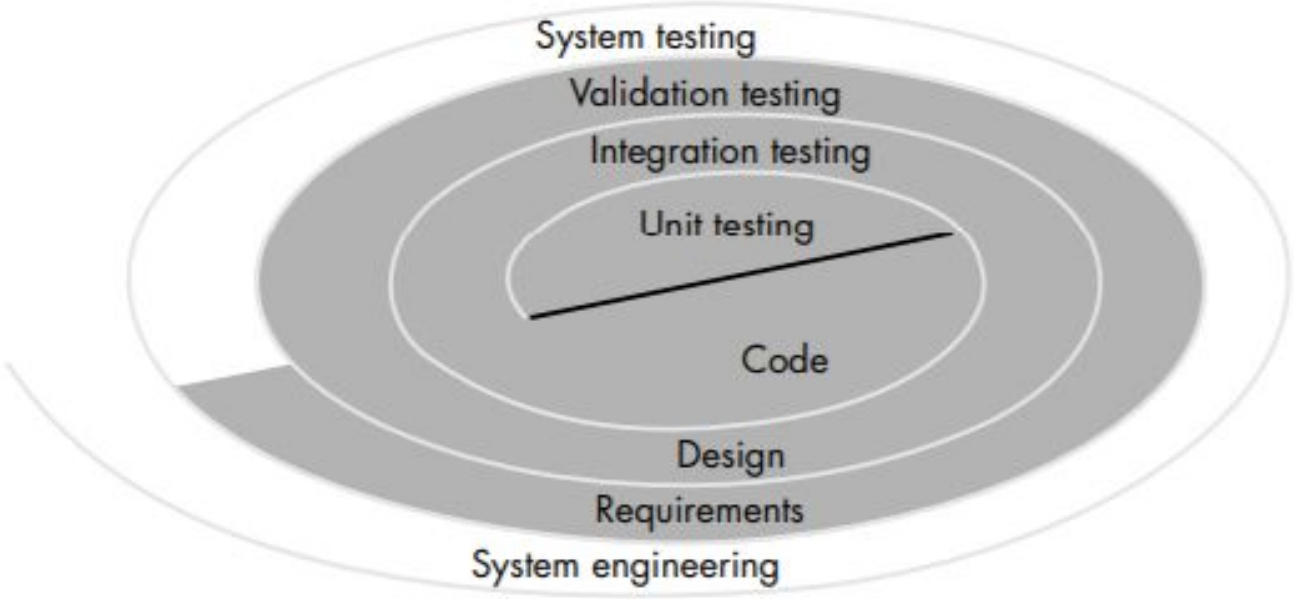
• Testing begins at the component level and works "outward" toward the integration of the entire computer-based system.

• Different testing techniques are appropriate 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.

Software Testing Strategy



Software Testing

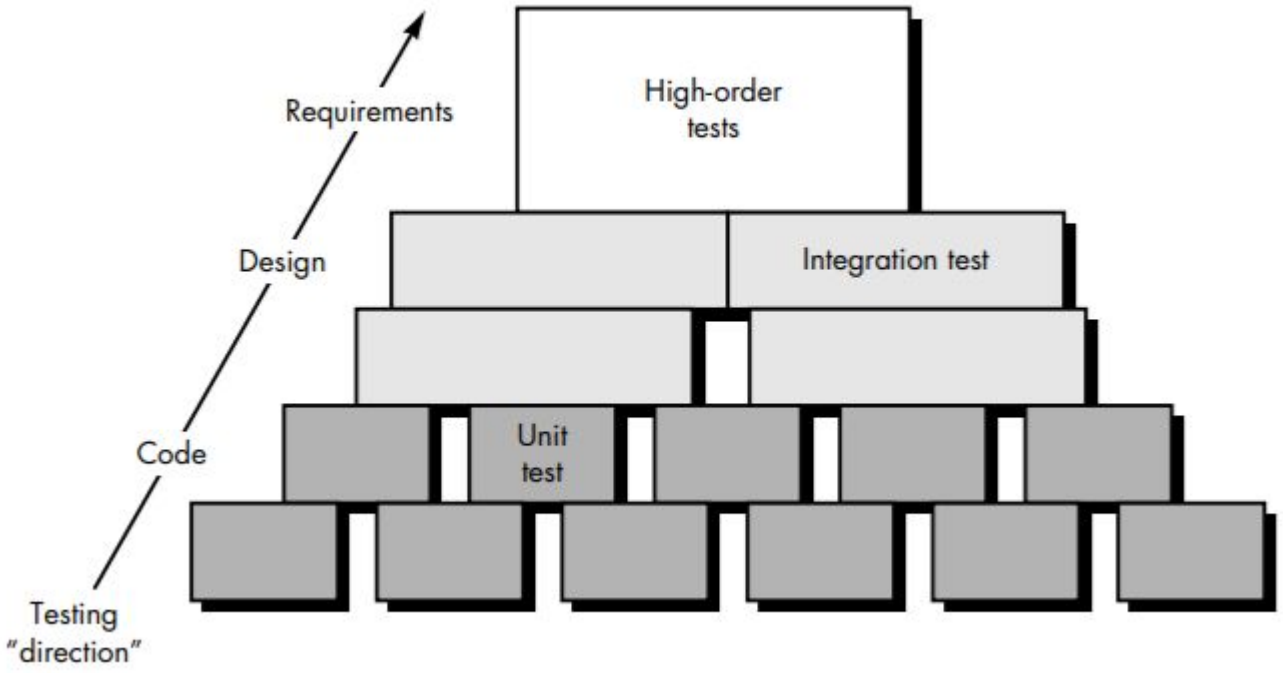
Unit testing makes heavy use of white-box testing techniques, exercising specific paths in a module's control structure to ensure complete coverage and maximum error detection.

Integration testing addresses the issues associated with the dual problems of verification and program construction. Black-box test case design techniques are the most prevalent during integration, although a limited amount of white-box testing may be used to ensure coverage of major control paths

Validation testing provides final assurance that software meets all functional, behavioral, and performance requirements. Black-box testing techniques are used exclusively during validation

System testing verifies that all elements mesh properly and that overall system function/performance is achieved.

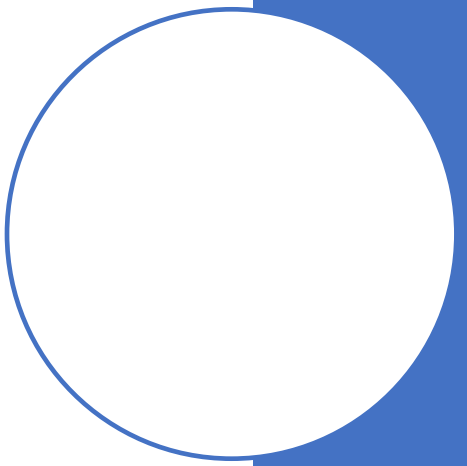
Software Testing



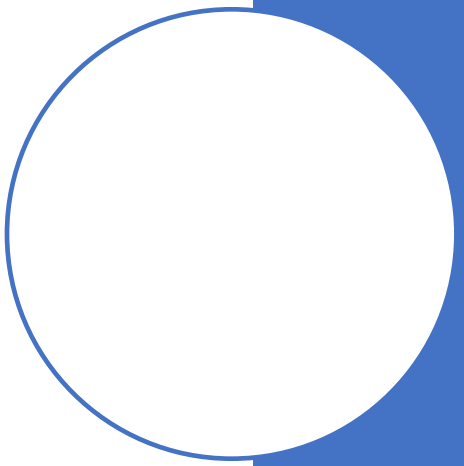
Criteria for Completion of Testing

One response to the question is: "You're never done testing; the burden simply shifts from you (the software engineer) to your customer. Every time the customer/user executes a computer program, the program is being tested.

Another response (somewhat cynical but nonetheless accurate) is: "You're done testing when you run out of time, or you run out of money."



Different Software Testing Strategies



Software Testing Techniques

White & Black box testing

Mutation testing: Mutation score

Unit Testing

Integration Testing

Alpha & Beta Testing

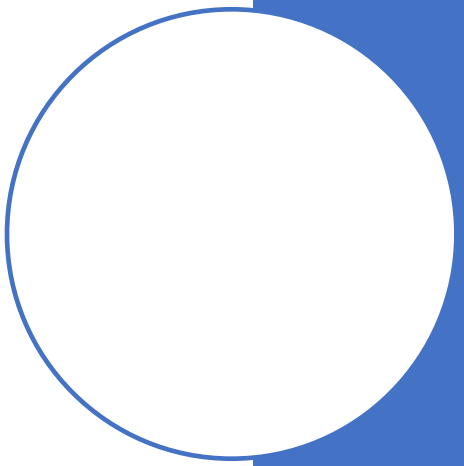
Recovery Testing

Security Testing

Stress Testing

Differential Testing.

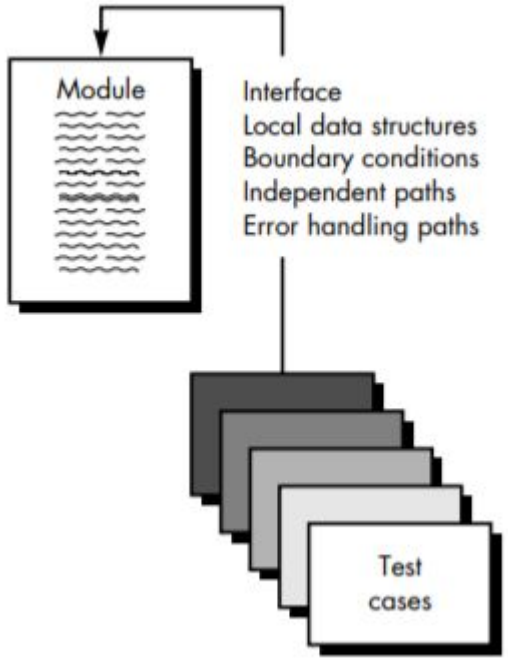
Adversarial Testing



Unit Testing

Important control paths are tested to uncover errors within the boundary of the module.

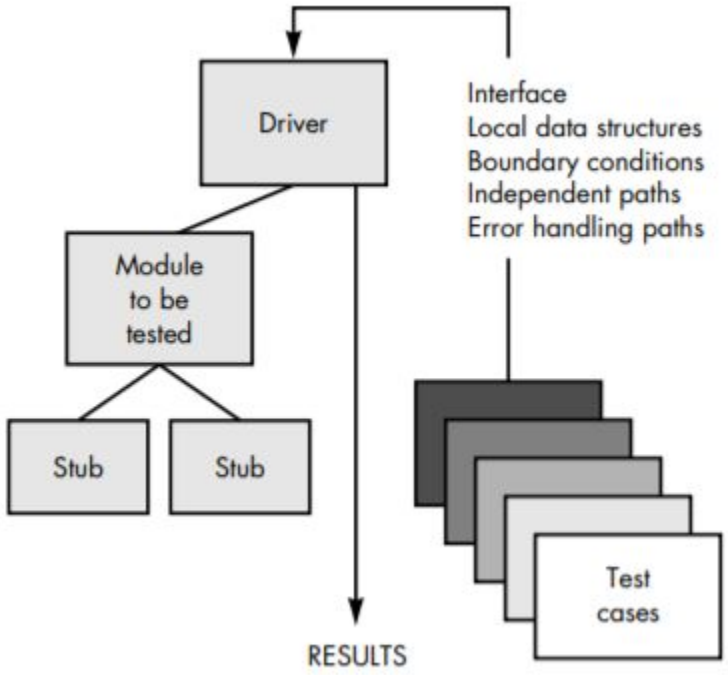
The unit test is white-box oriented, and the step can be conducted in parallel for multiple components



Unit Testing

Because a component is not a stand-alone program, driver and/or stub software must be developed for each unit test.

In actual environment, Driver and/or stub are not delivered with the final software product.



Integration Testing

Question: "If they all work individually, why do you doubt that they'll work when we put them together?

- The problem, of course, is "putting them together"—interfacing.
- Data can be lost across an interface.
- One module can have an inadvertent, adverse affect on another.
- Subfunctions, when combined, may not produce the desired major function.

Sadly, the list goes on and on….

1- Top-Down Integration
5 Steps Process

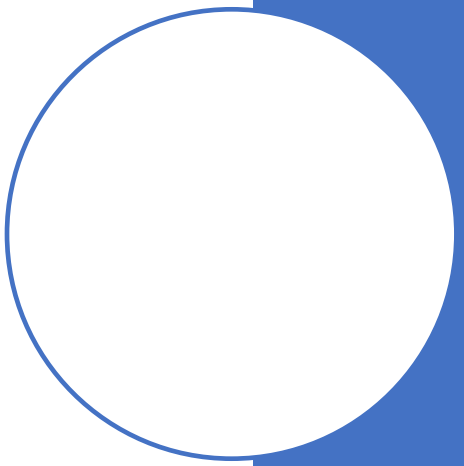
1. The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.

2. Depending on the integration approach selected (i.e., depth or breadth first), subordinate stubs are replaced one at a time with actual components.

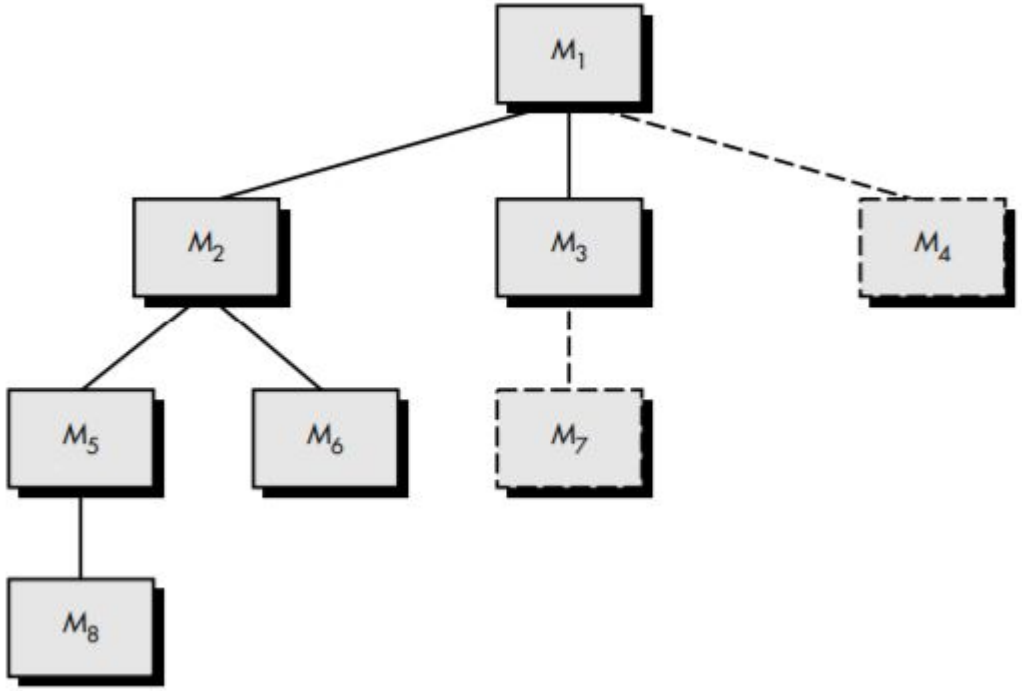
3. Tests are conducted as each component is integrated.

4. On completion of each set of tests, another stub is replaced with the real component.

5. Regression testing may be conducted to ensure that new errors have not been introduced



1- Top-Down Integration
Depth First / Breadth First



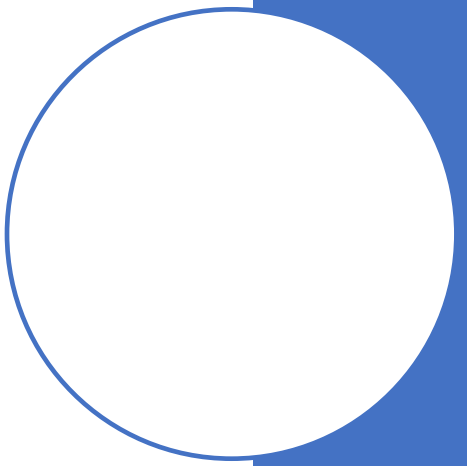
2- Bottom-up Integration
4 Steps Process

1. Low-level components are combined into clusters (sometimes called builds) that perform a specific software subfunction.

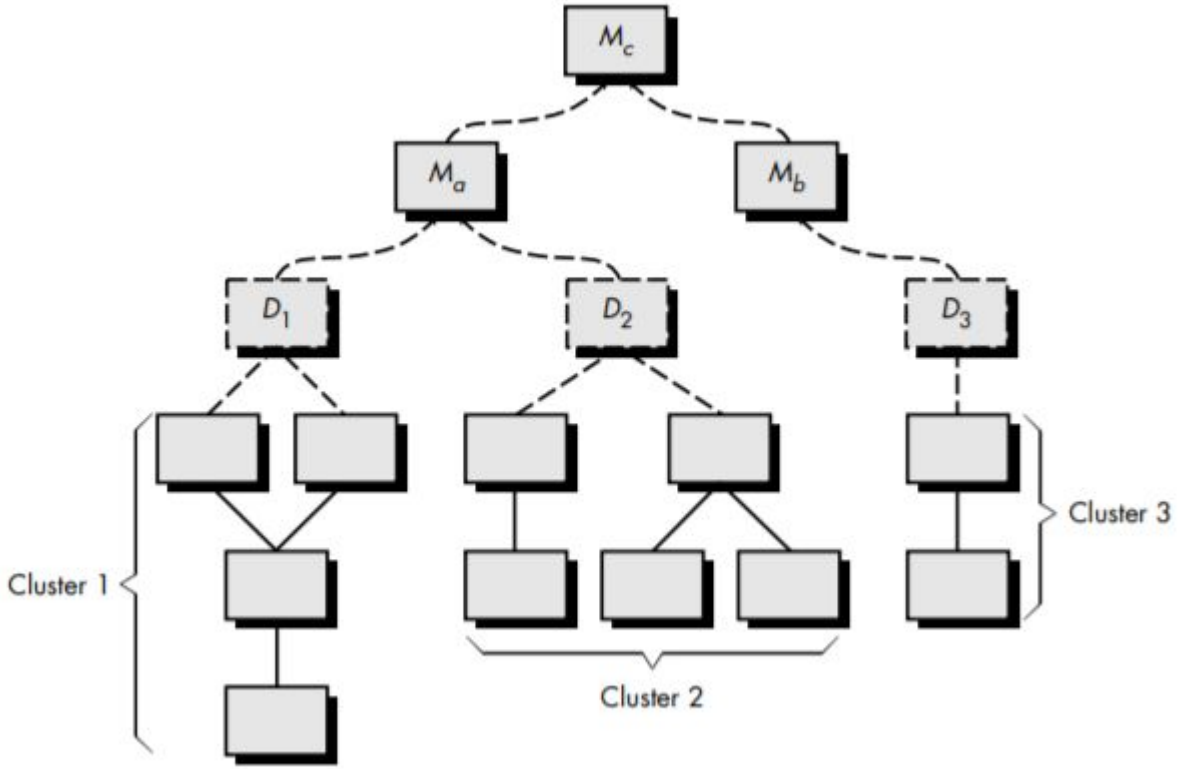
2. A driver (a control program for testing) is written to coordinate test case input and output.

3. The cluster is tested.

4. Drivers are removed and clusters are combined moving upward in the program structure.



2- Bottom-up Integration



Regression Testing

Each time a new module is added as part of integration testing, the software changes.

These changes may cause problems with functions that previously worked flawlessly.

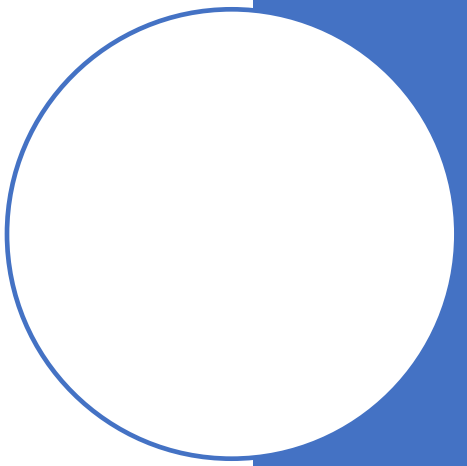
In the context of an integration test strategy, regression testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.

Therefore, the regression test suite should be designed to include only those tests that address one or more classes of errors in each of the major program functions.

Alpha & Beta Testing

The **alpha test** is conducted at the developer's site by a customer. The software is used in a natural setting with the developer "looking over the shoulder" of the user and recording errors and usage problems.

The **beta test** is conducted at one or more customer sites by the end-user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a "live" application of the software in an environment that cannot be controlled by the developer.

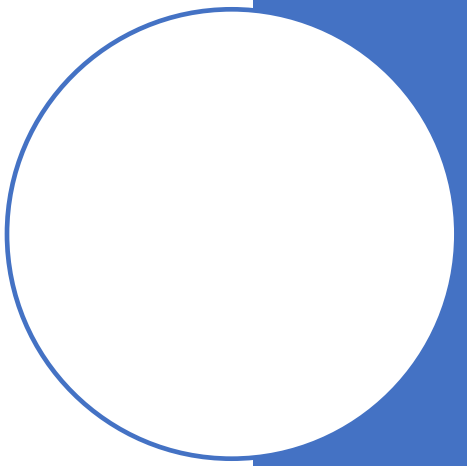


Recovery Testing

Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed.

If recovery is automatic (performed by the system itself), reinitialization, checkpointing mechanisms, data recovery, and restart are evaluated for correctness.

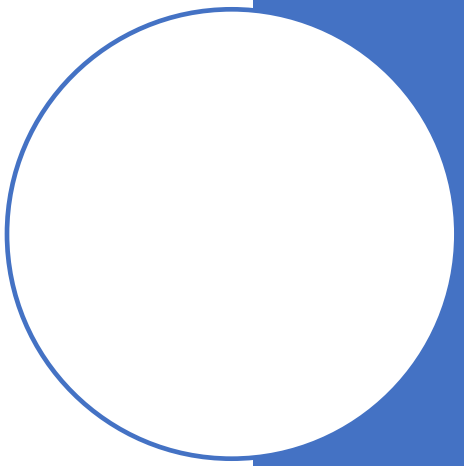
If recovery requires human intervention, the mean-time-to-repair (MTTR) is evaluated to determine whether it is within acceptable limits



Security Testing

Penetration spans a broad range of activities: hackers who attempt to penetrate systems for sport; disgruntled employees who attempt to penetrate for revenge; dishonest individuals who attempt to penetrate for illicit personal gain.

Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration.



Security Testing

During security testing, the tester plays the role(s) of the individual who desires to penetrate the system.

1. The tester may attempt to acquire passwords through external clerical means.

2. May attack the system with custom software designed to breakdown any defenses that have been constructed.

3. May overwhelm the system, thereby denying service to others;

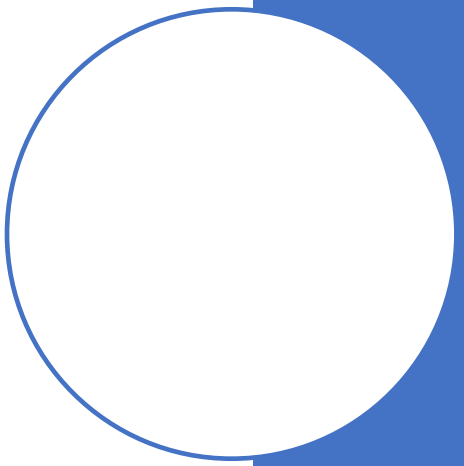
4. May browse through insecure data, hoping to find the key to system entry. As an example, using last name and age from one data source to find the sensitive information in some other linked data source.

Stress Testing

Stress tests are designed to confront programs with abnormal situations.

In essence, the tester who performs stress testing asks: "How high can we crank this up before it fails?“

Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency, or volume. As an example, (1) special tests may be designed that generate ten interrupts per second, when one or two is the average rate, (2) test cases that require maximum memory or other resources are executed, etc.



Debugging Process