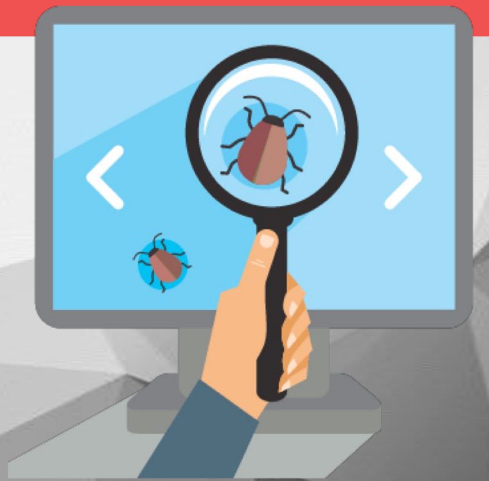
Unit 4.5
Software Engineering

Software Testing



Verification & Validation

Verification

Are we building the product right?

The objective of Verification is to make sure that the product being develop is as per the requirements and design specifications.

Validation

Are we building the right product?

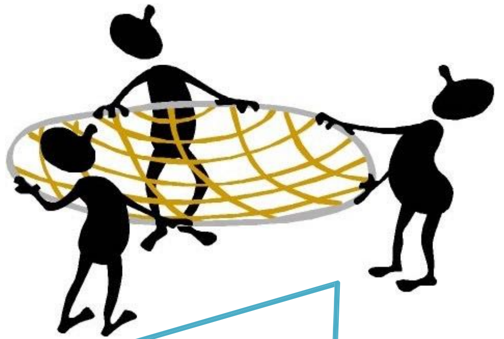
The objective of Validation is to make sure that the product actually meet up the user's requirements, and check whether the specifications were correct in the first place.

Verification vs Validation

Verification	Validation
Process of evaluating products of a development phase to find out whether they meet the specified requirements.	Process of evaluating software at the end of the development to determine whether software meets the customer expectations and requirements.
Activities involved: Reviews, Meetings and Inspections	Activities involved: Testing like black box testing, white box testing, gray box testing
Carried out by QA team	Carried out by testing team
Execution of code is not comes under Verification	Execution of code is comes under Validation
Explains whether the outputs are according to inputs or not	Describes whether the software is accepted by the user or not
Cost of errors caught is less	Cost of errors caught is high

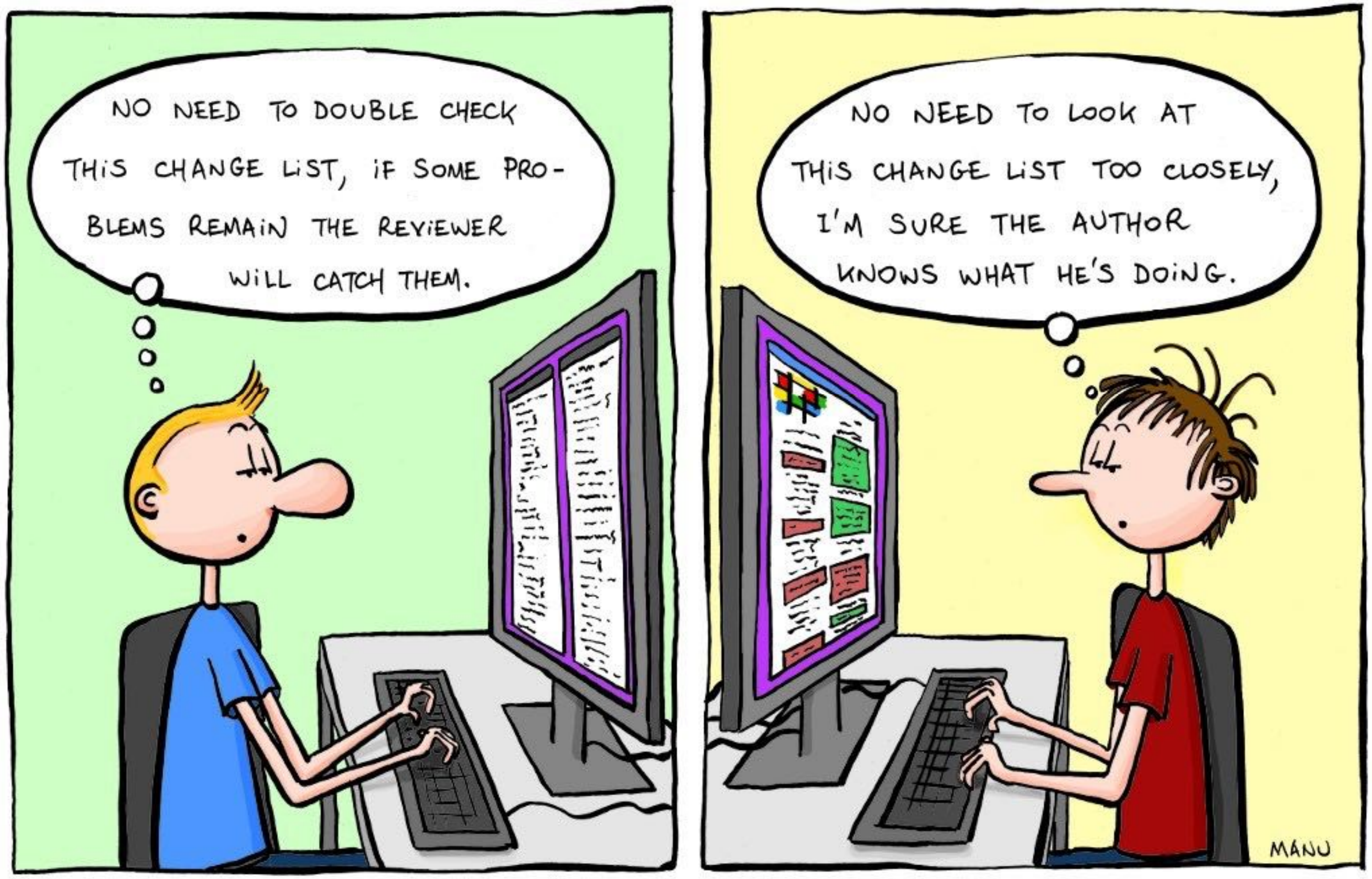
Software Testing

Testing is the process of exercising a program with the specific intent of finding errors prior to delivery to the end user.



Don't view testing as a "safety net" that will catch all errors that occurred because of weak software engineering practice.

Software Testing


NO NEED TO DOUBLE CHECK THIS CHANGE LIST, IF SOME PROBLEMS REMAIN THE REVIEWER WILL CATCH THEM.
NO NEED TO LOOK AT THIS CHANGE LIST TOO CLOSELY, I'M SURE THE AUTHOR KNOWS WHAT HE'S DOING.
MANU

Who Test the Software

Developer

Tester

Understands the system but, will test "gently" and, is driven by "delivery"

Must learn about the system, but, will attempt to break it and, is driven by quality

Testing without plan is of no point
It wastes time and effort

Testing need a strategy
Dev team needs to work with Test team, “Egoless Programming”

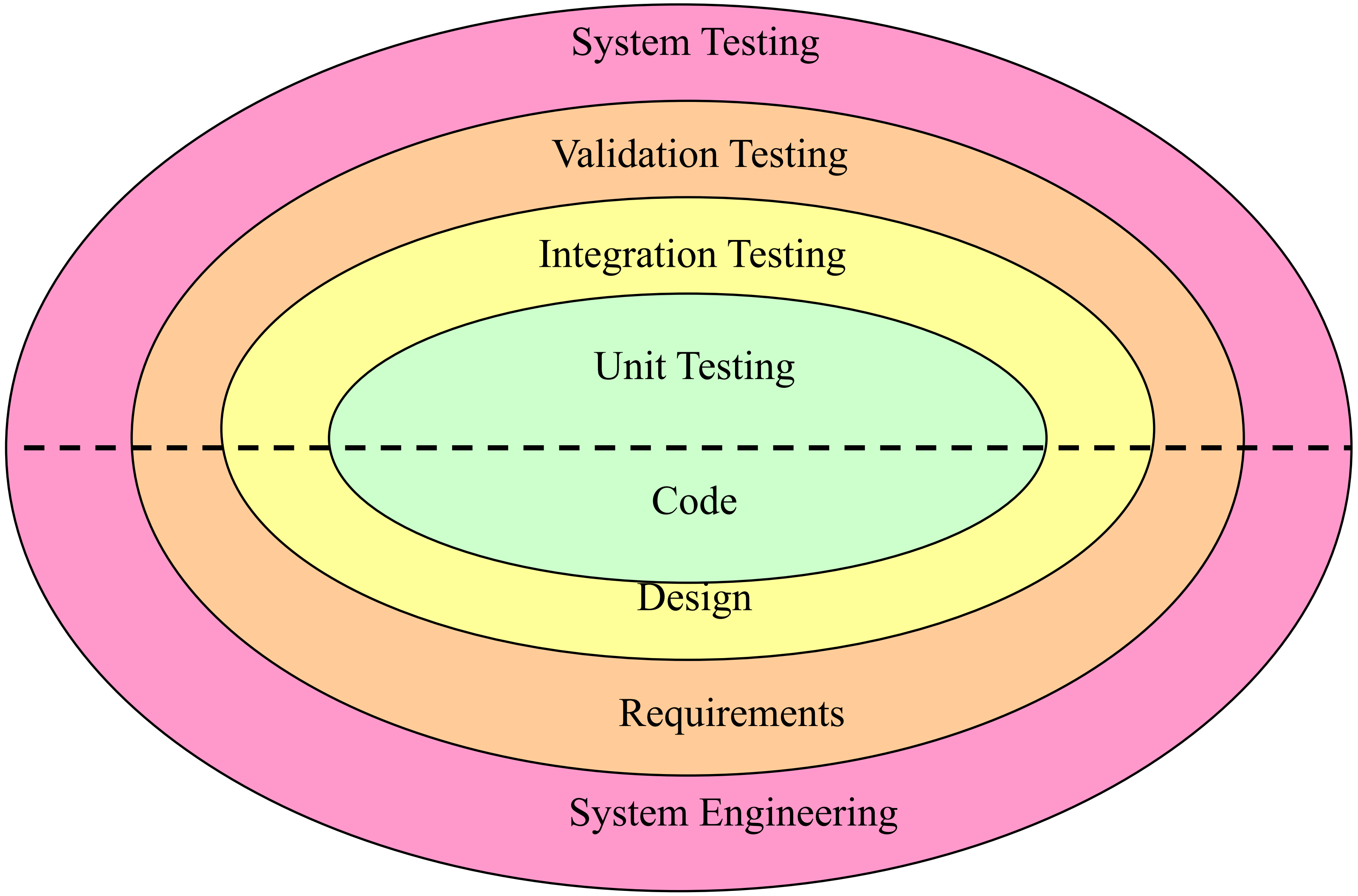
Software Testing Strategies

- A strategy for software testing integrates the design of software test cases into a well-planned series of steps that result in successful development of the software
- The strategy provides a road map that describes the steps to be taken, when, and how much effort, time, and resources will be required
- The strategy incorporates test planning, test case design, test execution, and test result collection and evaluation
- Because of time pressures, progress must be measurable and problems must surface as early as possible

Characteristics of Strategic Testing

- To perform effective testing, a software team should conduct effective formal technical reviews
- Testing begins at the component level and work 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 by an independent test group
- Testing and debugging are different activities, but debugging must be accommodated in any testing strategy

Software Testing Strategy



Software Testing Strategy Cont.

Unit Testing

- It concentrate on each unit of the software as implemented in source code.
- It focuses on each component individual, ensuring that it functions properly as a unit.

Integration Testing

- It focus is on design and construction of software architecture
- Integration testing is the process of testing the interface between two software units or modules

Software Testing Strategy Cont.

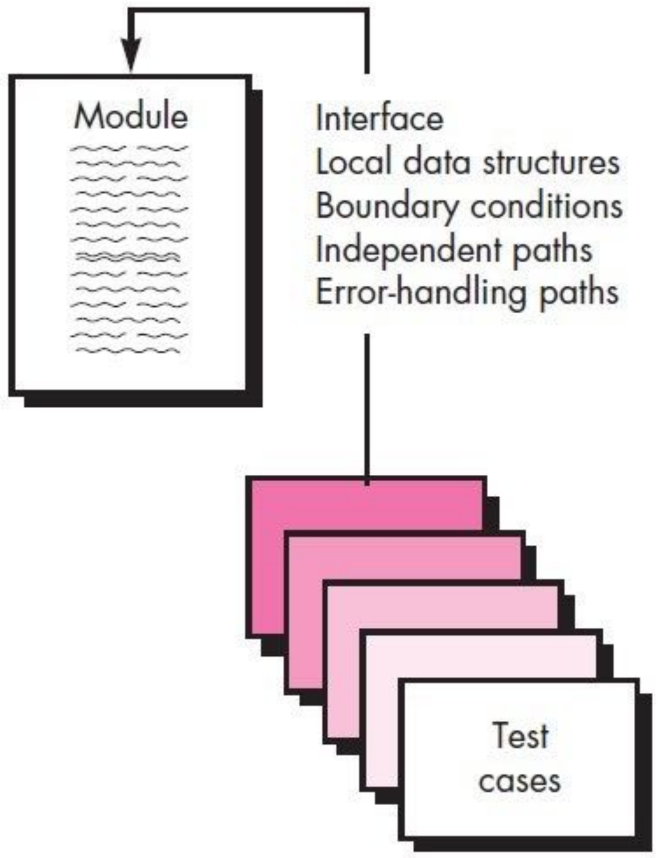
Validation Testing

- Software is validated against requirements established as a part of requirement modeling
- It give assurance that software meets all informational, functional, behavioral and performance requirements

System Testing

- The software and other software elements are tested as a whole
- Software once validated, must be combined with other system elements e.g. hardware, people, database etc…
- It verifies that all elements mesh properly and that overall system function / performance is achieved.

Unit Testing

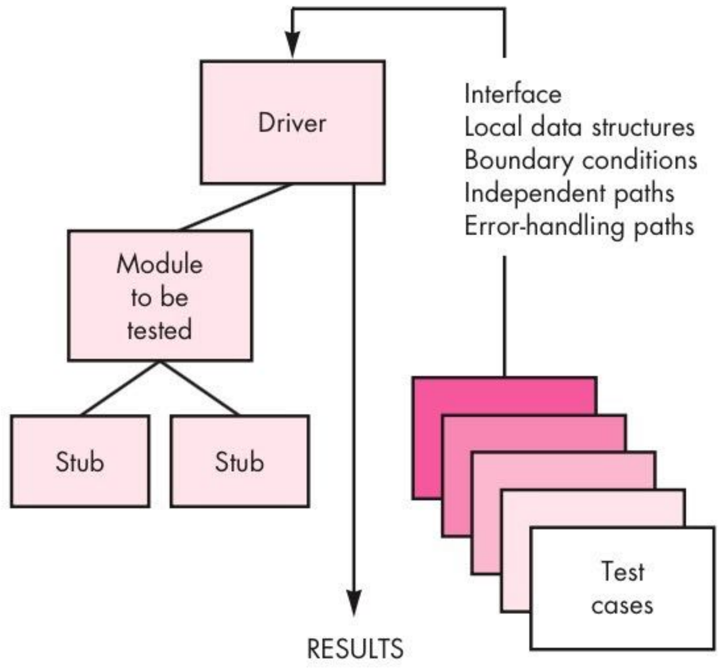


- Unit is the smallest part of a software system which is testable.
- Unit Testing validates small building block of a complex system before testing an integrated large module or whole system
- The unit test focuses on the internal processing logic and data structures within the boundaries of a component.

Unit Testing Cont.

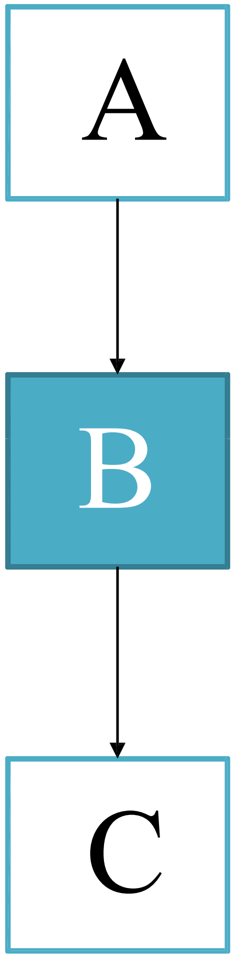
- The module is tested to ensure that information properly flows into and out of the program unit
- Local data structures are examined to ensure that data stored temporarily maintains its integrity during execution
- All independent paths through the control structures are exercised to ensure that all statements in module have been executed at least once
- Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restricted processing
- All error handling paths are tested

Unit Testing Cont.



- Component-testing (Unit Testing) may be done in isolation from rest of the system
- In such case the missing software is replaced by Stubs and Drivers and simulate the interface between the software components in a simple manner

Unit Testing Cont.



- Let's take an example to understand it in a better way.
- Suppose there is an application consisting of three modules say, module A, module B & module C.
- Developer has design in such a way that module B depends on module A & module C depends on module B
- The developer has developed the module B and now wanted to test it.
- But the module A and module C has not been developed yet.
- In that case to test the module B completely we can replace the module A by Driver and module C by stub

Unit Testing Cont.

- Driver and/or Stub software must be developed for each unit test
- A driver is nothing more than a "main program"
  - It accepts test case data
  - Passes such data to the component and
  - Prints relevant results.
- Driver
  - Used in Bottom up approach
  - Lowest modules are tested first.
  - Simulates the higher level of components
  - Dummy program for Higher level component

Unit Testing Cont.

- Stubs serve to replace modules that are subordinate (called by) the component to be tested.
- A stub or "dummy subprogram"
  - Uses the subordinate module's interface
  - May do minimal data manipulation
  - Prints verification of entry and
  - Returns control to the module undergoing testing
- Stubs
  - Used in Top down approach
  - Top most module is tested first
  - Simulates the lower level of components
  - Dummy program of lower level components