CHPATER SEVEN
Testing

A STRATEGIC APPROACH TO SOFTWARE TESTING

Verification and Validation

Verification: "**Are we building the product right**?“

Validation: "**Are we building the right product**?”

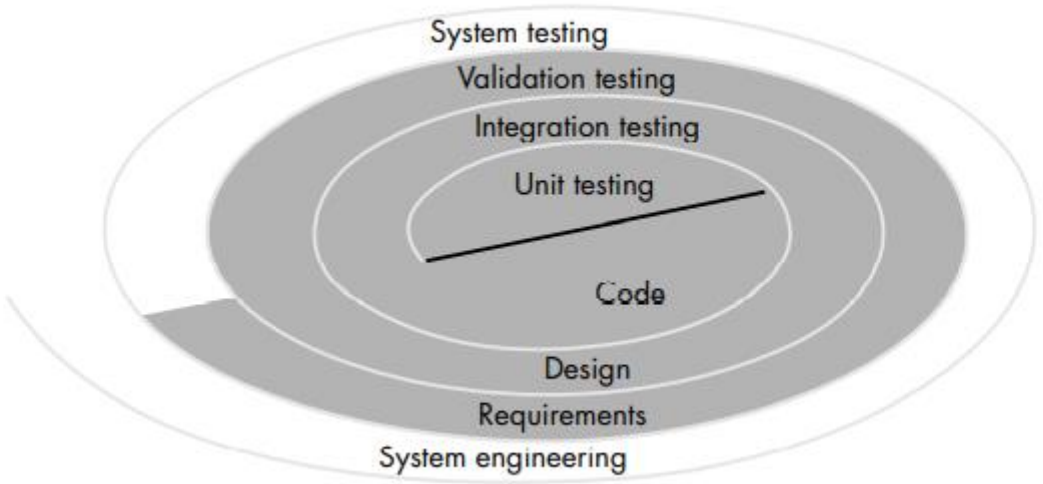
- Verification refers to the set of activities that ensure that software correctly implements a **specific function**.
- Validation refers to a different set of activities that ensure that the software that has been built is traceable to **customer requirements.**

A STRATEGIC APPROACH TO SOFTWARE TESTING

- Verification and validation includes a wide array of **SQA activities**
- V & V include **formal technical reviews**, quality and configuration audits, performance monitoring, simulation, feasibility study, documentation review, database review, algorithm analysis, development testing, qualification testing.

A Software Testing Strategy

FIGURE 18.1
Testing strategy



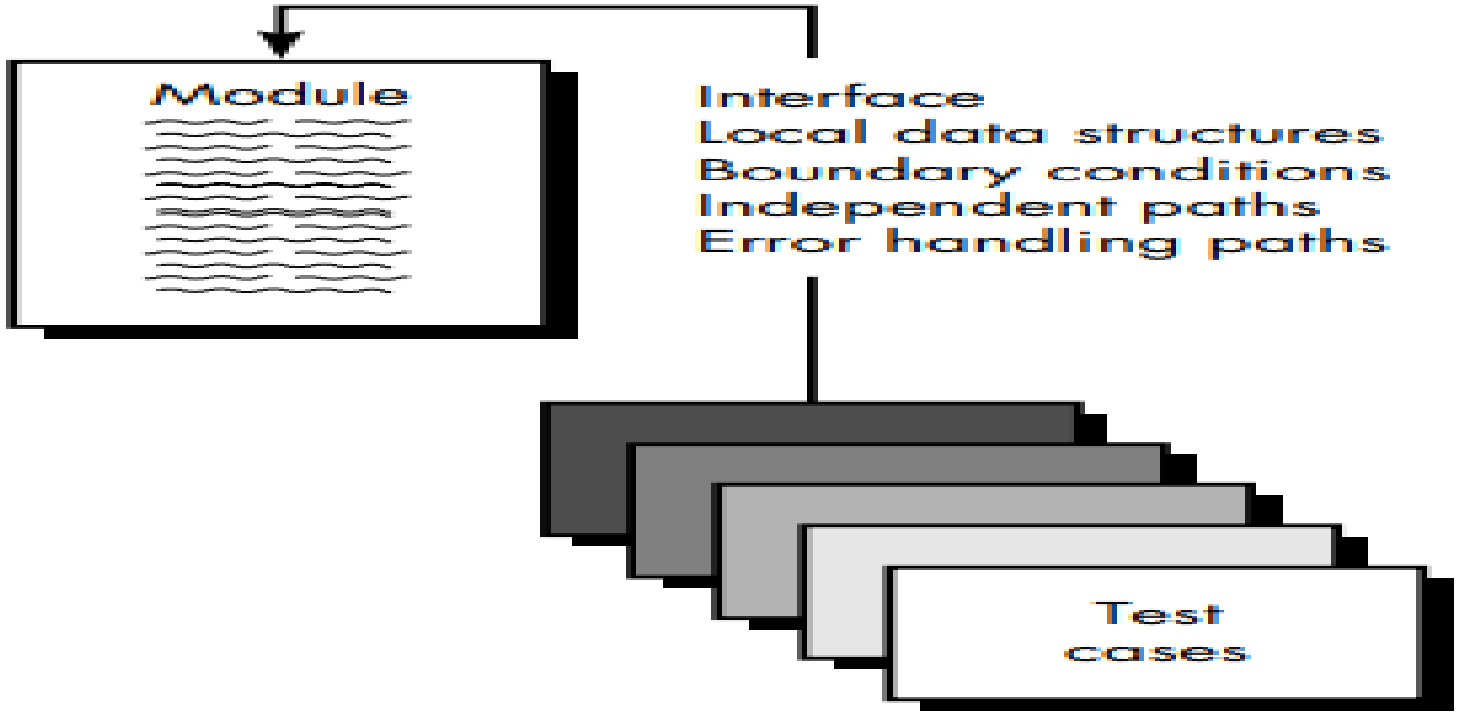
A Software Testing Strategy

- The software engineering process may be viewed as the spiral
- 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.
- While moving inward along the spiral, we come to design and finally to coding

Unit Testing

- Unit testing focuses verification effort on the **smallest unit** of software design.
- Control paths are tested to uncover errors within the boundary of the module

Unit Test Considerations



Unit Test Consideration

- The module interface is tested to ensure that information properly flows into and out of the program unit under test.
- The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution.
- Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
- All independent paths (basis paths) through the control structure are exercised to ensure that all statements in a module have been executed at least once.
- And finally, all error handling paths are tested

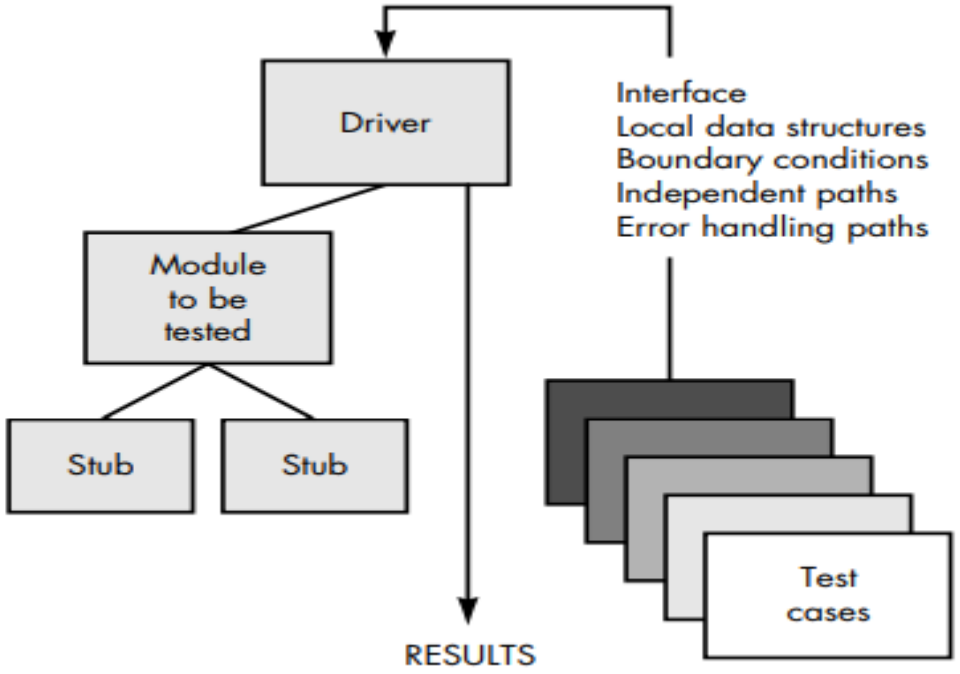
Unit Test Procedure

- Unit testing is normally considered as an adjunct to the **coding step**.
- After source level code has been developed, reviewed, and verified for correspondence to component level design, unit test case design begins.

Unit Test Procedure

- Driver and/or stub software is developed for each unit test.
- In most applications a driver is nothing more than a "main program" that accepts **test case data**, passes such data to the component (to be tested), and **prints relevant results**.
- A stub or "dummy subprogram" uses the subordinate module's interface.
- Stubs serve to replace modules that are subordinate the component to be tested.

Unit Test Procedure



Integration Testing

- 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 adverse affect on another*
    - *sub functions, when combined, may not produce the desired major function;*
    - *individually acceptable error may be magnified to unacceptable levels;*
    - *global data structures can present problems.*

Integration Testing

- **Non-incremental integration :** follow big bang approach → the entire program is tested as a whole.
- **Incremental Integration :**Program is constructed and tested in small increments

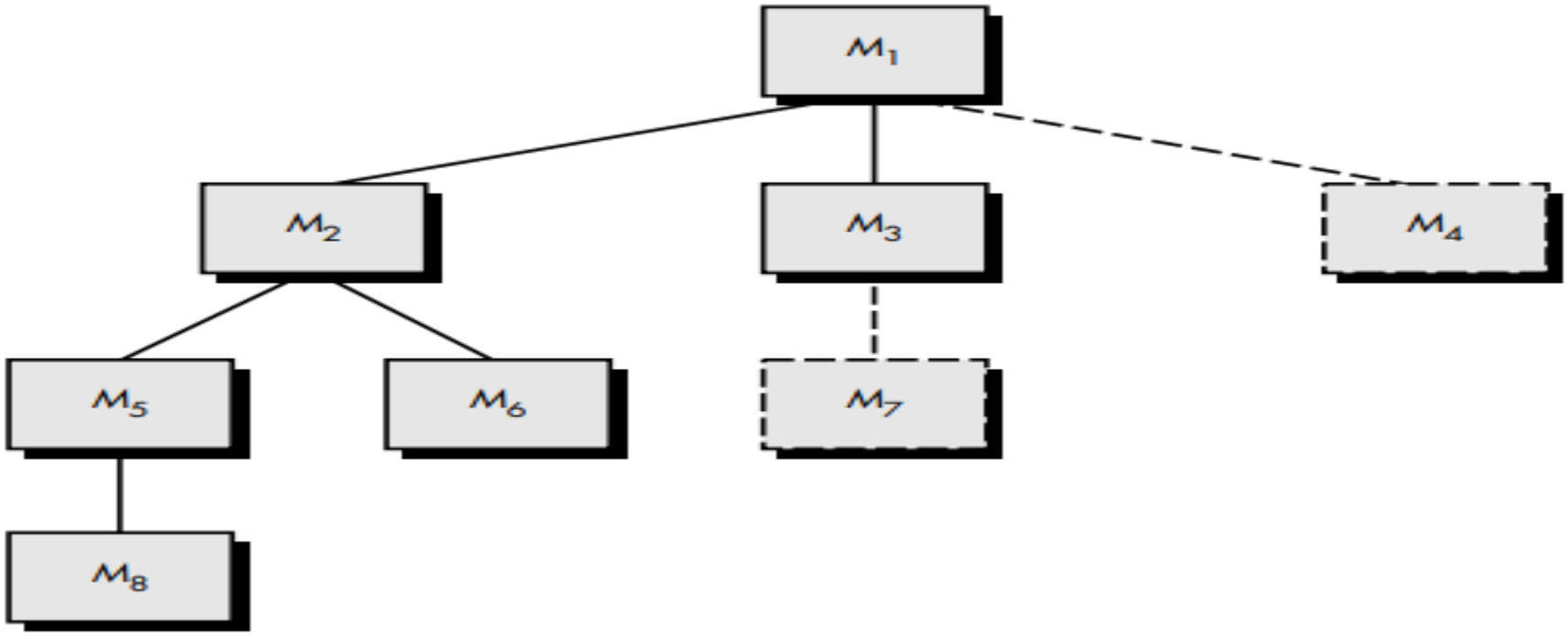
Integration Testing

1. **Top Down Integration**
2. **Bottom-Up Integration**

Top-Down Integration

- Top-down integration testing is an **incremental approach** to construction of program structure.
- Modules are integrated by moving **downward through the control** hierarchy, beginning with the main control module (main program).
- Modules subordinate to the main control module are incorporated into the structure in either a depth-first or breadth-first manner

Top-Down Integration



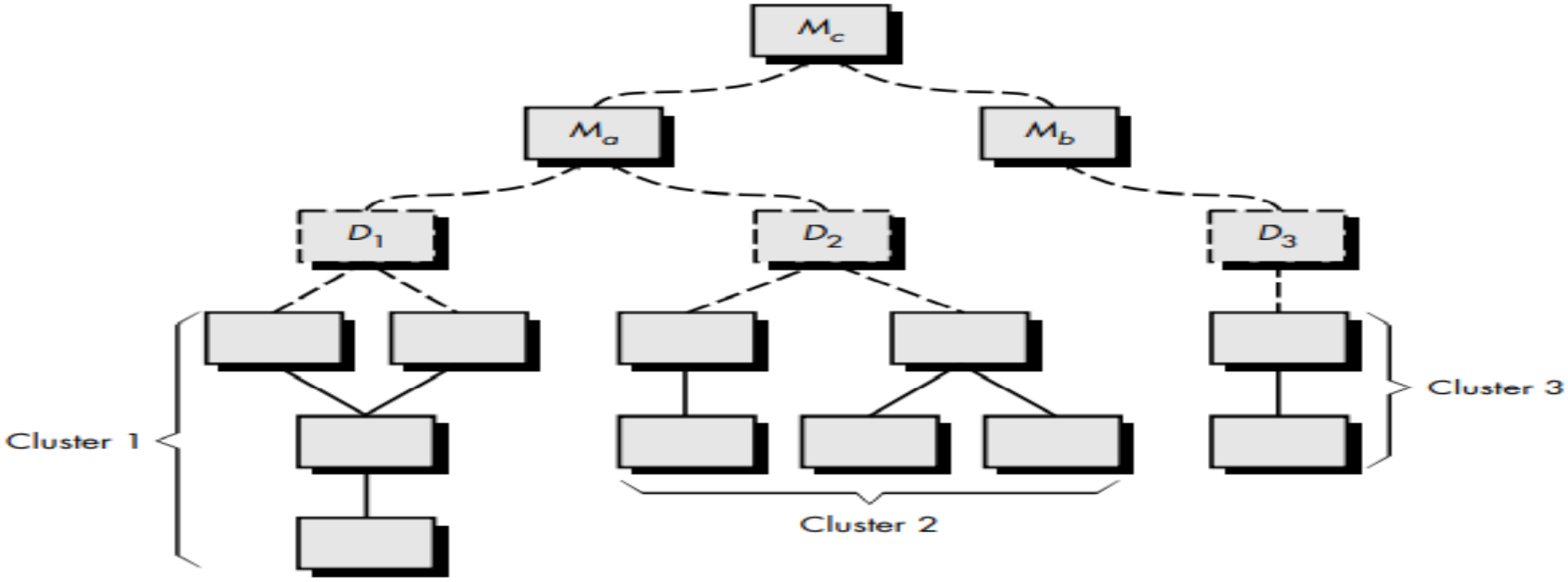
Top-Down Integration

- **Depth- first Integration** – M1, M2, M5, M8
- **Breath-first Integration** –Components M2,M3 and M4 would be integrated first. The next control level M5, M6

Bottom-up Integration

- Components are combined to form clusters 1, 2, and 3.
- Each of the clusters is tested using a driver (shown as a dashed block). Components in clusters 1 and 2 are subordinate to Ma.
- Drivers D1 and D2 are removed and the clusters are interfaced directly to Ma.
- Similarly, driver D3 for cluster 3 is removed prior to integration with module Mb. Both Ma and Mb will ultimately be integrated with component Mc, and so forth

Bottom-up Integration



Regression Testing

- Each time a new module is added as part of integration testing, the software changes.
- New data flow paths are established, new I/O may occur, and new control logic is invoked.
- 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.

Regression Testing

- Regression testing may be conducted **manually**, by **re-executing a subset of all test cases** or using **automated capture/playback tools**.
- **Capture/playback** tools enable the software engineer to capture test cases and results for subsequent playback and comparison.

- Automatic – Software use
- Manual – We manually write the test cases

SMOKE TESTING

- Smoke testing is one of the popular software testing services performed after software **build** to find the **functionalities** of the program are working fine.

Smoke testing approach encompasses the following activities:

1. *Software components that have been translated into code are integrated into a "**build.**"* A build includes all data files, libraries, reusable modules, and engineered components that are required to implement one or more product functions**.**

2. *A series of tests is designed to **expose errors***

3. *The build is **integrated** with other builds and the entire product (in its current form) is smoke tested daily*.

SMOKE TESTING

- Smoke testing provides a number of benefits:
- *Integration risk is minimized*
- *The quality of the end-product is improved*
- *Error diagnosis and correction are simplified*
- *Progress is easier to assess*

SMOKE TESTING

VALIDATION TESTING

- Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by the customer

VALIDATION TESTING

- After each validation test case has been conducted, one of two possible conditions exist:

(1) The function conform to s**pecification** and are accepted

(2) A deviation from specification is **uncovered** and a **deficiency** list is created

Configuration Review

- An important element of the validation process is a configuration review.
- The intent of the review is to ensure that all **elements** of the software **configuration** have been properly developed, are cataloged, and have the necessary detail to support phase of the software life cycle.
- The configuration review, sometimes called an **audit**

Alpha and Beta Testing

Alpha Testing

- The alpha test is conducted at the **developer's site** by a **customer.**
- The developers are **presented** during testing
- The developers records the errors and usage problems
- Alpha tests are conducted in a **controlled environment.**

Alpha and Beta Testing

Beta Testing

- Conducted at the **customer sites** by the **end users**
- The customer records all the errors and reports them to the developer at regular intervals
- Software is used in an **uncontrolled environment**.

System Testing

- System testing is actually a series of different tests whose primary purpose is to **fully exercise** the computer-based system.

System Testing

1. *Recovering Testing --*

2.Security Testing--

3. Stress Testing – ()

4. Performance Testing--- ()

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** re-initialization, 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

- Security testing attempts to **verify** that protection mechanisms built into a system will, in fact, protect it from improper penetration.
- During security testing the tester plays the role of **hackers**

Stress Testing

- Stress testing are designed to confront programs with **abnormal situation.**
- Stress testing executes a system in a manner that **demands resources** in **abnormal** quantity, frequency or volume
- A **variation** of stress testing is a technique called **sensitivity testing**.

Performance Testing

- Performance testing is designed to test the **run-time performance** of software within the context of an integrated system.
- Performance testing occurs throughout **all steps** in the testing process.
- Even at the unit level, the performance of an individual module may be assessed as **white-box tests** are conducted.
- Performance tests are often coupled with **stress testing** and usually require both hardware and software instrumentation.