Chapter 17

## Software Testing Strategies

Testing strategy must incorporate test planning, test case design, test execution, and resultant data collection and evaluation.

**What is it?**

Software is tested to uncover errors that were made inadvertently as it was designed and constructed.

**Who does it?**

Developed by the project manager, software engineers, and testing specialists.

**What are the steps?**

Testing begins “in the small” and progresses “to the large.”

**What is the work product?**

A Test Specification documents the software team’s approach to testing by defining a plan that describes an overall strategy and a procedure that defines specific testing steps and the types of tests that will be conducted.

**How do I ensure that I’ve done it right?**

By reviewing the Test Specification prior to testing.

**17.1 A STRATEGIC APPROACH TO SOFTWARE TESTING**

Testing is a set of activities that can be planned in advance and conducted systematically. Testing stratigies must have the following generic characteristics:

- Effective technical reviews
- Begins at the component level and works “outward” toward the integration
- Conducted by the developer of the software and (for large projects) an independent test group
- Testing and debugging are different activities

A strategy for software testing must accommodate low-level tests as well as high level tests.

### 17.1.1 Verification and Validation

- 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 [Boe81] states this another way:

- Verification: “Are we building the product right?”
- Validation: “Are we building the right product?”

Verification and validation includes SQA activities:

- Technical reviews

- Quality and configuration audits
- Performance monitoring
- Simulation
- Feasibility study
- Documentation review
- Database review
- Algorithm analysis
- Development testing, usability testing, qualification testing, acceptance testing, and installation testing.

### 17.1.2 Organizing for Software Testing

Developers have a vested interest in demonstrating that the program is error-free, that it works according to customer requirements, and that it will be completed on schedule and within budget. So it should be tested through unbiased person (**independent test group (ITG)**). There are often a number of misconceptions

(1) The developer of software should do no testing at all,

(2) The software should be "tossed over the wall" to strangers who will test it mercilessly,

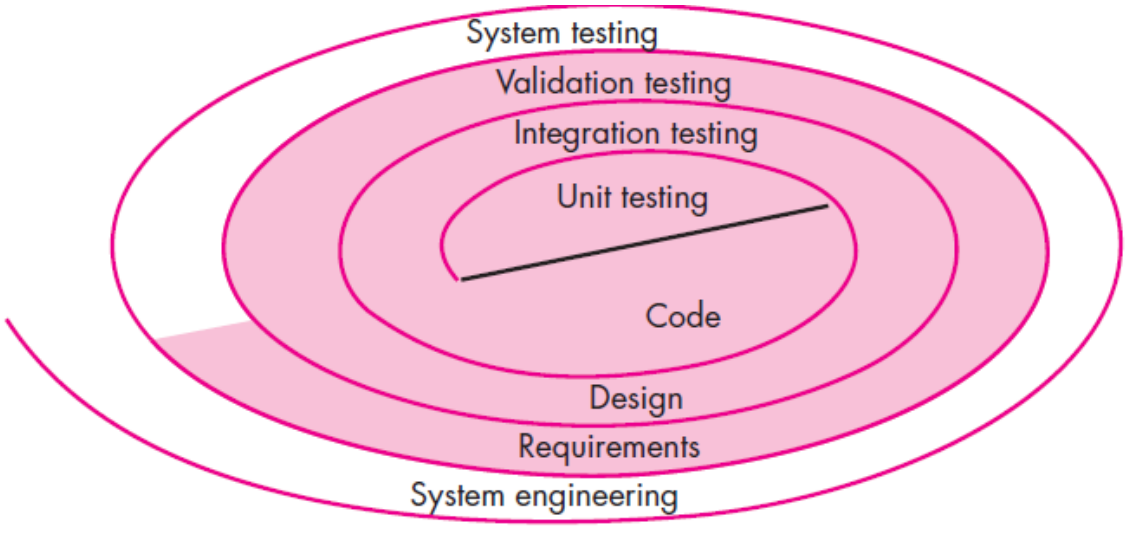
(3) Testers get involved with the project only when the testing steps are about to begin.

The software developer is always responsible for testing the individual units.

Independent testing removes the conflict of interest that may otherwise be present. After all, ITG personnel are paid to find errors. The developer and the ITG work closely throughout a software project to ensure that thorough tests will be conducted. While testing is conducted, the developer must be available to correct errors that are uncovered. **ITG reports to the software quality assurance organization, thereby achieving a degree of independence.**

### 17.1.3 Software Testing Strategy—The Big Picture

The software process may be viewed as the spiral illustrated in Figure 17.1.




Reference:- R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill, Ed 7, 2010.

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. Moving inward along the spiral, you come to design and finally to coding.

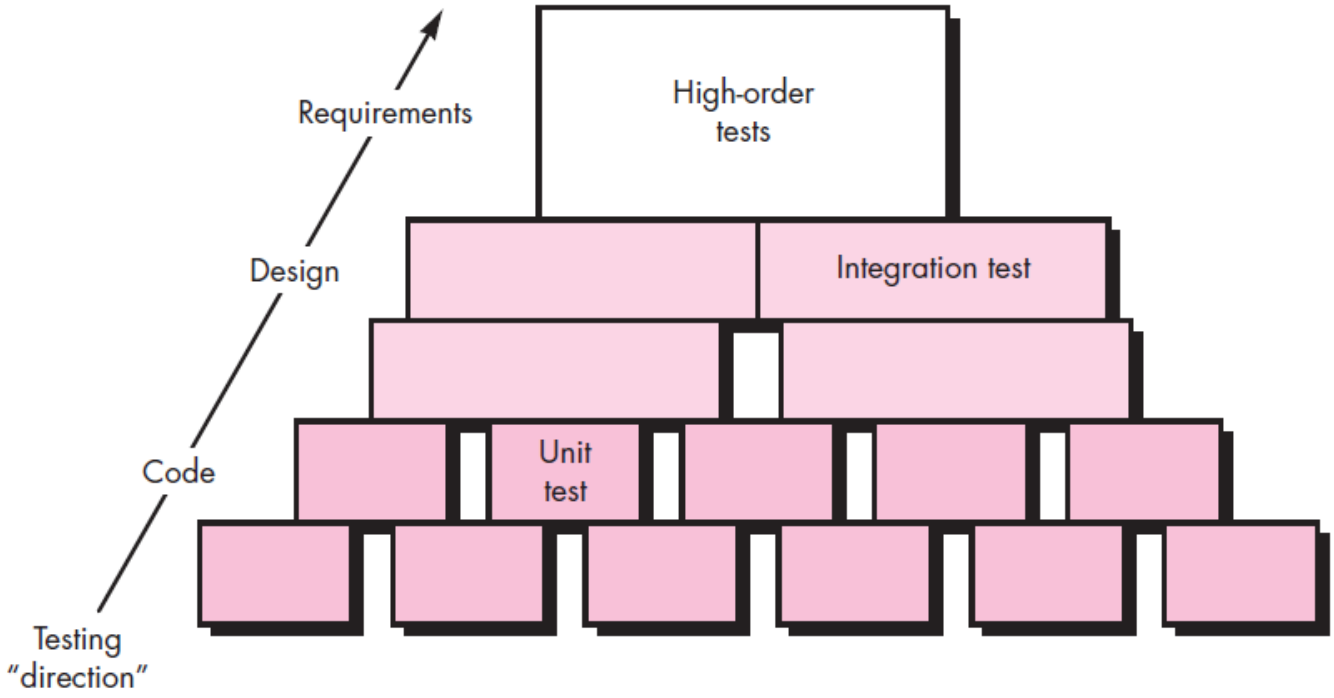
**Unit testing** begins at the vortex of the spiral and concentrates on each unit. Unit testing makes heavy use of testing techniques that exercise specific paths in a component's control structure to ensure complete coverage and maximum error detection.

**Integration testing,** where the focus is on design and the construction of the software architecture. Test case design techniques that focus on inputs and outputs are more prevalent during integration.

**Validation testing**, where requirements established as part of requirements modeling are validated against the software that has been constructed.

**System testing,** where the software and other system elements are tested as a whole.

**Testing within the context of software engineering is actually a series of four steps that are implemented sequentially.**




Reference:- R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill, Ed 7, 2010.