Công Cụ & Phương Pháp Thiết Kế - Quản Lý (Phần Mềm)

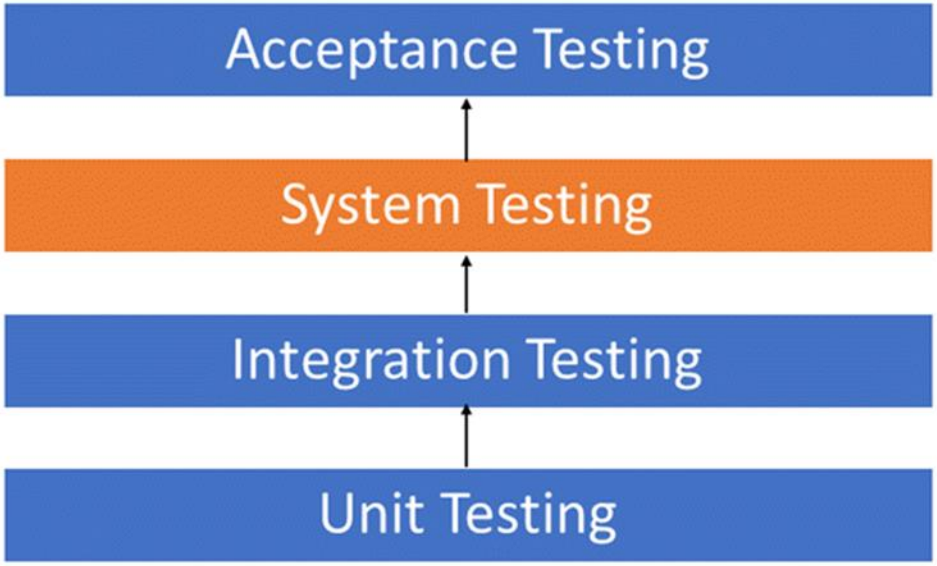
TRAN KIM SANH
Instructor of DTU

Email: trankimsanh@dtu.edu.vn
Tel: 0987 409 464

Testing and Quality

Contents

- Is Quality an Accident?
- Test Driven Development
- Integration Test
- Acceptance Test
- System Test
- Test Coverage



Is Quality an Accident?

- Quality is never an accident; it is always the result of intelligent effort." - John Ruskin

Who Owns Quality?

You Do!

Characteristics of Software Quality

- Correctness
  - Defects
- Usability
- Performance
- Scalability
- Extensibility

- Installability
- Maintainability*
- Portability*
- Reusability*
- Readability*
- Testability*

** Internal software characteristic*

Characteristics of Software Quality

- Accessability
- Auditability
- Configurability
  - Personalization
  - Internationalization
- Efficiency
- Interoperability
- Operational availability
- Flexibility*
- Reliability
- Robustness
- Safety
- Security

* Internal software characteristic

Quality Approaches

- Find defects
- Prevent defects
- Do Both

Development
Research
Analysis
Baseline System
Prototype
Design
Production
Testing
Planing
Release to Market
Support
Deliver
Software Quality Assurance

"Testing can only prove the presence of features and defects, not the absence of defects"

"You can't test in quality"

Defect Detection Rates - 1

Removal Step	Lowest Rate	Modal Rate	Highest Rate
Informal design reviews	25%	35%	40%
Formal design inspection	45%	55%	65%
Informal code reviews	20%	25%	35%
Formal code inspections	45%	60%	70%
Modeling or prototyping	35%	65%	80%
Personal desk checking of code	20%	40%	60%

Steve McConnell, Code Complete $_{2}^{nd}$ *edition. page 470*

Defect Detection Rates - 2

Removal Step	Lowest Rate	Modal Rate	Highest Rate
Unit test	15%	30%	50%
Integration test	25%	35%	40%
Regression test	15%	25%	30%
System test	25%	40%	55%
Low-volume beta test (<10)	25%	35%	40%
High-volume beta test (>10)	60%	75%	85%

- Different steps find different defect types
- Combine steps to find the most defects

Steve McConnell, Code Complete $_2^{nd}$ edition. page 470

Defect Detection Rates - 3

Removal Step	Lowest Rate	Modal Rate	Highest Rate
Informal design reviews	25%	35%	40%
Informal code reviews	20%	25%	35%
Personal desk checking of code	20%	40%	60%
Unit test	15%	30%	50%
Integration test	25%	35%	40%
Regression test	15%	25%	30%
Expected Cumulative defect-removal efficiency	~74%	~90%	~97%

Steve McConnell, Code Complete $_2{}^{nd}$ edition. page 470

Quality – General Principles

- Defects creep into software at all stages
- Improving quality reduces development time and costs

> "The best way to improve productivity and quality is to reduce the time spent reworking code"

Steve McConnell, Code Complete $_{2}$nd edition. page 470

Quality Costs

- Quality is not free, so what does it cost?
- Studies have found that defect removal is the most expensive and time consuming work on software projects
    - Up to 50% for poorly run projects
- Most studies have found that inspections are cheaper than testing
- Finding defects earlier reduces their costs

Steve McConnell, Code Complete $_2$nd edition. page 470

Quality Plan – What to Do?

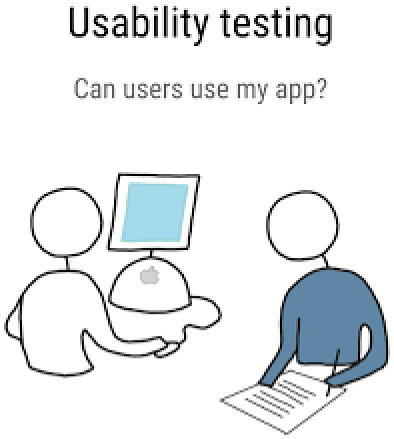
- Consider an integrated approach:
  - Inspections of all requirements, architecture and designs for critical parts of a system
  - Modeling or prototyping
  - Code inspections (lightweight)
  - Test driven development
  - Continuous integration
    - ✓ Automated builds
    - ✓ Automated unit test
  - Integration test
  - Systems test (including performance, etc.)

Quality Plan – What to Do?

- Remember that you need to prevent defects as well as detect them:
  - Inspections help cross-train the team on the system, tools, programming style, etc.
  - Techniques such as test driven development can help build in more systematic testing
  - Build and test automation used in continuous integration can help ensure that what was working yesterday still works today

Quality Plan – What to Do?

- Remember to add other elements, as necessary:
  - □ Inspections of all requirements, architecture and designs for critical parts of a system
  - □ Usability testing
  - □ Modeling or prototyping
  - □ Code inspections (lightweight)
  - □ Test driven development
  - □ Continuous integration
  - □ Integration test
  - □ Systems test (including performance, etc.)



Quality Plan – What to Do?

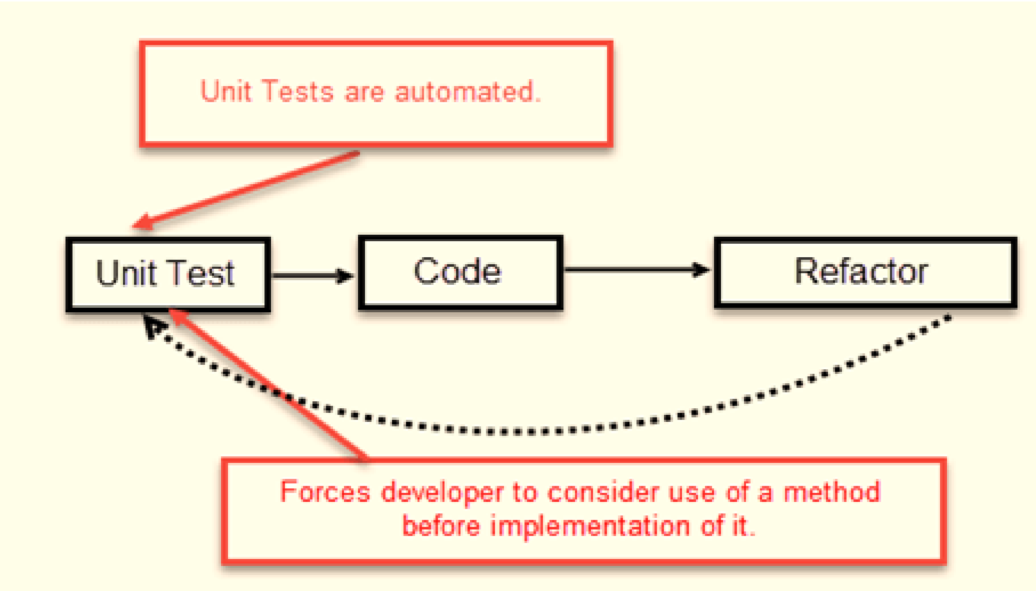
- Quality costs will also increase if the product demands a higher degree of reliability
  - Medical systems
  - Airplane guidance systems
  - Weapons systems

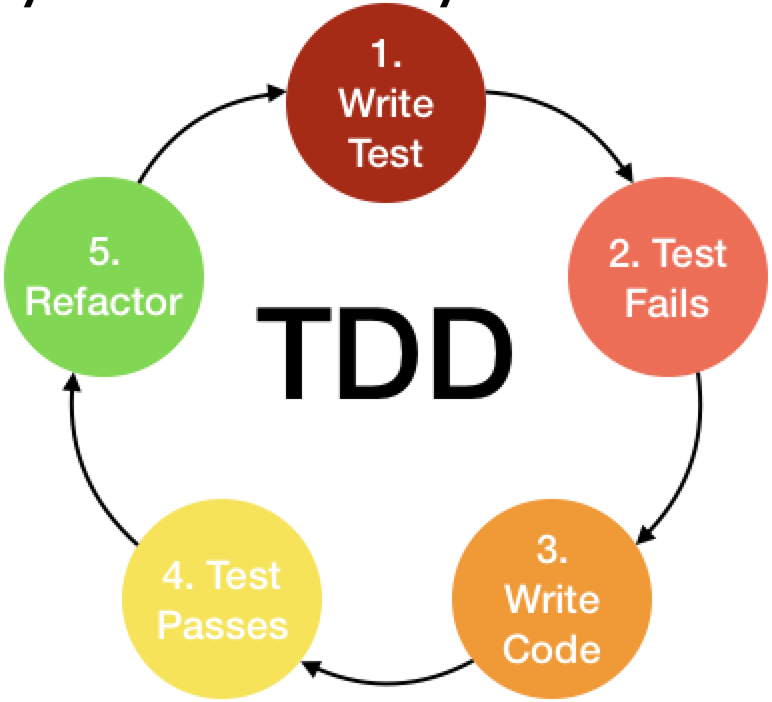
Testing

- Testing cannot be expected to catch every error in the program - it is impossible to evaluate all execution paths for all but the most trivial programs
- The goal of unit testing is to isolate each part of the program and show that the individual parts are correct
- We also need to catch integration errors, or broader system level errors (such as functions performed across multiple units, or non-functional test areas such as performance)

Test Driven Development

- Testing approach matters
  - Test all at once at the end
- vs
  - Test continuously and systematically


Unit Tests are automated.
Unit Test
Code
Refactor
Forces developer to consider use of a method before implementation of it.



Referenced from Prof.Redley of CMU

Test Driven Development

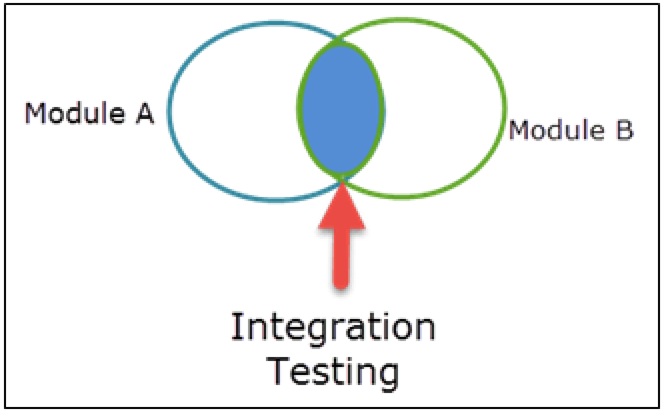
- Test-driven development requires developers to create automated unit tests that define code requirements before writing the code itself
- The unit tests contain assertions that are either true or false
- Passing the tests confirms correct behavior as developers evolve and refactor the code

Test Driven Development

- Evolve and expand your test suite over time
  - Whenever a defect is found
  - When adding new features
  - When you look at someone else's code and see a testing hole
- Test all of your code for functionality
  - If you miss something a test for it will be added eventually
- Over time you will have a collection of tests that will verify functionality at the unit level

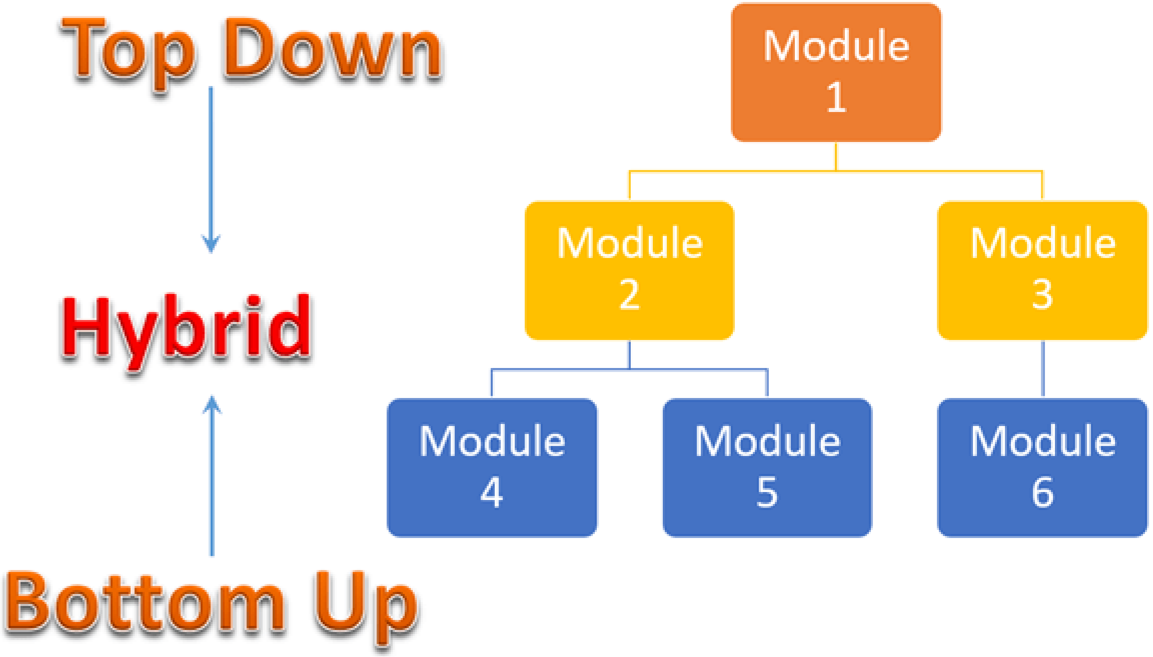
What is Integration Testing?

- Verifies that:
  - Components interact correctly
  - The interaction results are consistent with the function requirement for those components
- Emphasizes exercising the interfaces between components



Int. Testing Alternatives

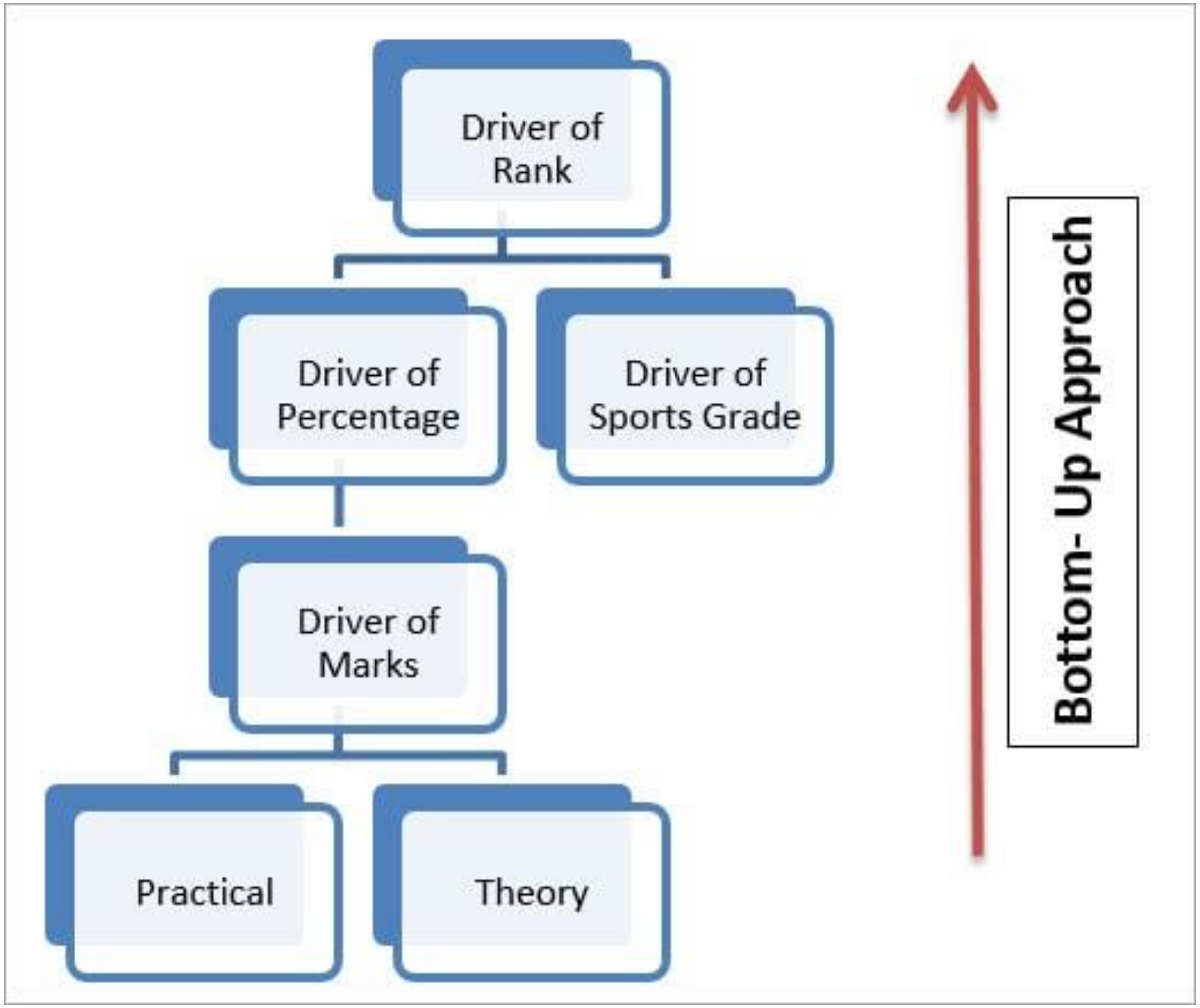
- Bottom-up
- Top-down
  - Depth-first
  - Breadth-first
- Hybrid



Int. Testing: Bottom-Up

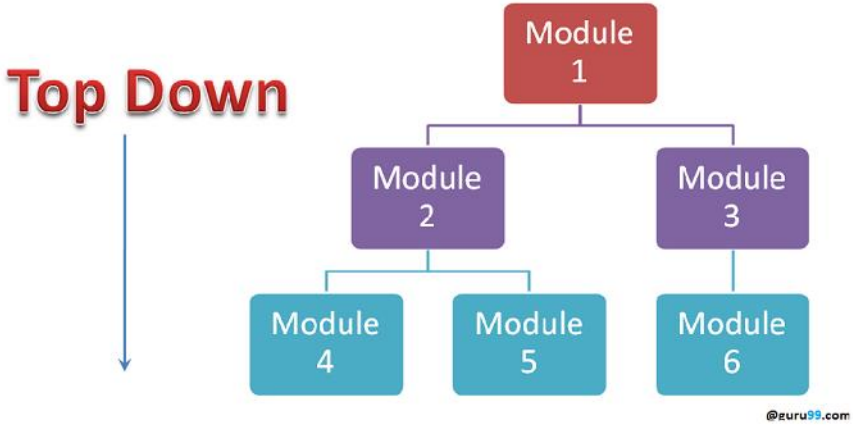
- Start after unit testing
- Identify modules
  - Combination of units that work/interface together
  - Focus on the feature being implemented
  - Identify how this feature impacts other areas
- Test each module
- Repeat this process progressively to higher levels

Int. Testing: Bottom-Up



Integration Testing: Top-Down

- Begin testing at the highest level first
- Progressively identify next lower level modules
- Run tests at this new level
- Two basic approaches to successive level
  - Depth first
  - Breadth first

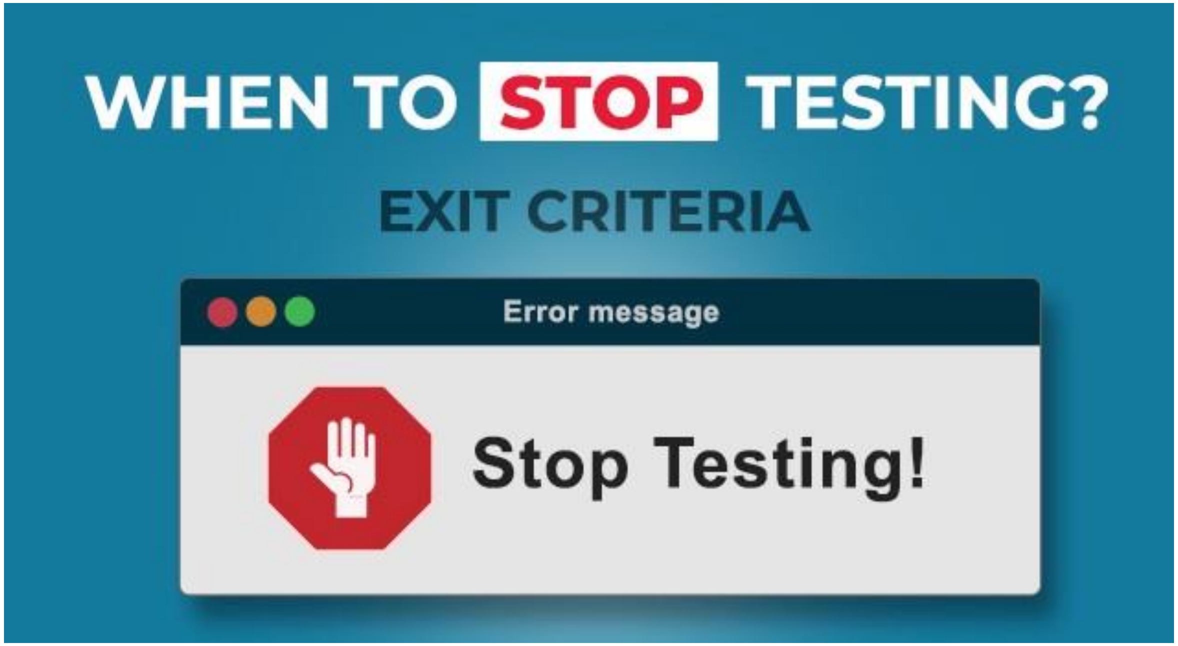


Integration Testing: Hybrid

- Used in comprehensive software development environment
- Bottom up: usually done first
  - Takes the component view
  - New components interface correctly
- Top down: usually done last
  - Takes the product feature view
  - System view of new feature

Int. Testing: Exit criteria

- All interfaces where components interact are successfully tested
- Negative test cases should also be covered



Integration Testing Issues

- Tedious and time consuming
- How much is enough?

Acceptance Testing

- Does the system satisfy the user's/ customer's acceptance criteria?
  - Be sure to define acceptance criteria at project start
- Most testing performed by developers & testers
- User/customer has acceptance testing responsibility
- Is the user's/customer's responsibility
  - Many users/customers don't know how to perform acceptance testing and will need help

Acceptance Testing

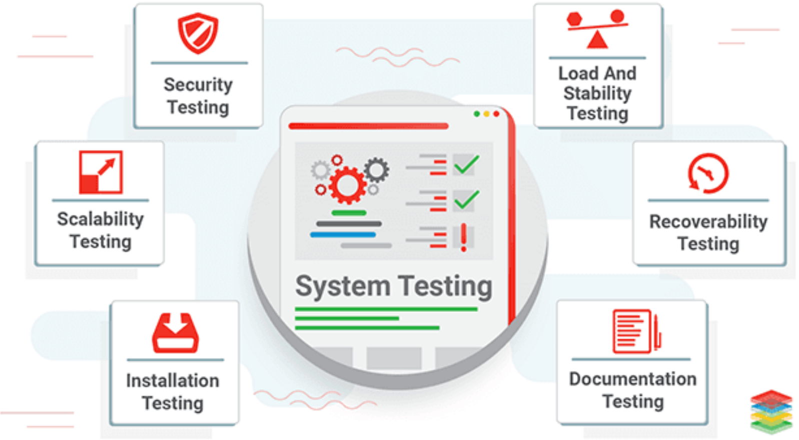
- Traditional acceptance testing
  - □ Alpha testing:
    - ✓ End user testing in a somewhat controlled test environment
    - ✓ “Friendly” environment
  - □ Beta testing:
    - ✓ End user testing at end user's site
    - ✓ Full range of environments
- Agile acceptance testing
  - □ At the end of each sprint
  - □ Not as thorough as system test
    - ✓ Verify that product functions as desired

Acceptance Testing - 2

- End user actively involved in acceptance test plan definition
  - Identify the criteria which, when met, will cause the customer to accept the product
  - Usually a joint effort between development & user/customer
- Acceptance test plan components
  - Acceptance criteria and schedule for all deliverables
  - Acceptance activities and who will perform

Systems Testing

- Follows unit, integration & acceptance testing
- Performs a variety of tests
  - □ "black box" functional testing
  - □ Nonfunctional
  - □ Performance
  - □ Load
  - □ Stress
  - □ Scalability
- …



Test Coverage

- There is no way today to say: "there are absolutely no defects”
- Being able to state that "we've executed every line of code, and all our tests passed"
  - Adds to our comfort level
  - Still doesn’t allow us to say “there are absolutely no defects”
- Defining the tests (and therefore test coverage) still requires human thought

Group discussion?

- Find defects in Login form

Summary

- Testing is just one part of a quality plan
- Be sure to plan for your quality activities:
  - Processes/practices
  - Deliverables
  - Learning curve
  - Time
  - Effort
  - Resources
- $
- Prevent defects as well as detect them

Video link

- Seven Testing Principles: Software Testing
- How to write a TEST CASE? Software Testing Tutorial

References

- ❑ Ian Sommerville. *Software engineering update 10th edition.* Wesley Computer Publishing 2018 Page: 226- 241