SOFTWARE TESTING

Program Testing:

- Intentions:
  - To verify program gives intended behaviour
  - check for defects, errors, anomalies
- Testing is done using artificial data
- Guides regarding non-functional attributes of system

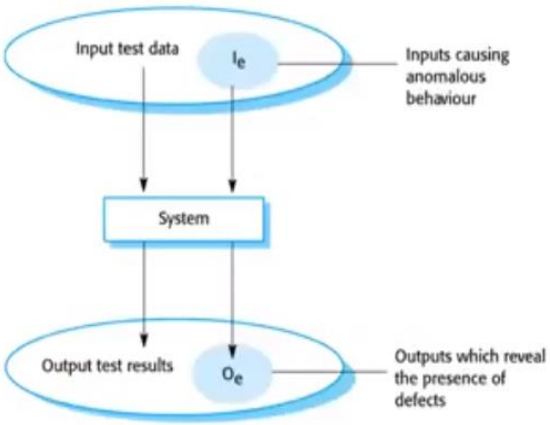
Testing goals:

- Validation testing goals:
  - To demonstrate to the developer or customer that software meets requirements. For customer-based every requirement should be tested, while for generic software, all features are tested
- Defect testing goals:
  - To expose bugs or point where the software does not conform specification. Should confirm that the errors should be handled and displayed in user-friendly fashion

Validation and defect testing:

- Validation testing: System functionalities are validated using a variety of test-cases to see system behaviour complies with requirements
- Defect testing: testing for defects by intentionally providing wrong data to system to check for sytem crashes, wrong computations, data corruptions etc.

Input output model (program testing:)



- Black box testing: where the QA or user is unaware of all the system specifications (hidden from user)

Verification vs Validation:

- Verification:
    - Checking whether product is made properly, according to specification
    - to resolve any software defect, requirement verify, inspection, review
- Validation:
    - Check whether the correct product was made
    - Complies with end-user requirements and correct output given

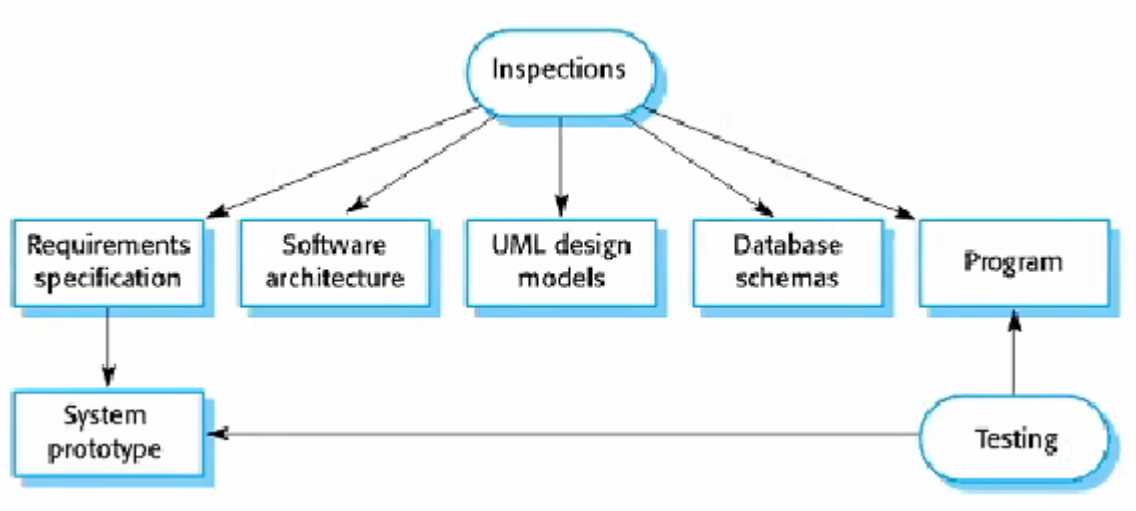
V & V confidence:

- Confidence that software is ‘fit for purpose’
- Factors:
    - Software purpose: how much critical is the software to the organizaiton (error may cause damage or not)
    - User expectation: user may not be ‘that much’ concerned with proper development of software
    - Marketing environment: releasing product early to acquire market due to high competition rather than fixing defects (e.g airlift vs swvl)

Inspections and testing:

- Inspection for smaller system is less preferred and testing is used to v & v the software. More likely for large critical systems, however recommended for improved quality
- Static verification: analysis of software based on system representation, documentation. Reviewed by customers or small inspections team made from within the organization. Example code walkthrough, peer reviews
- Dynamic verification: analysis based on working software to analyze system behaviour practically and assess operational behaviour
- Inspection and testing are complementary so defects missed in inspection are covered in testing.

- Inspection checks in conformance with specifications as working software is not available hence doesn't account for non-functional requirements
- Both are used during V & V process.



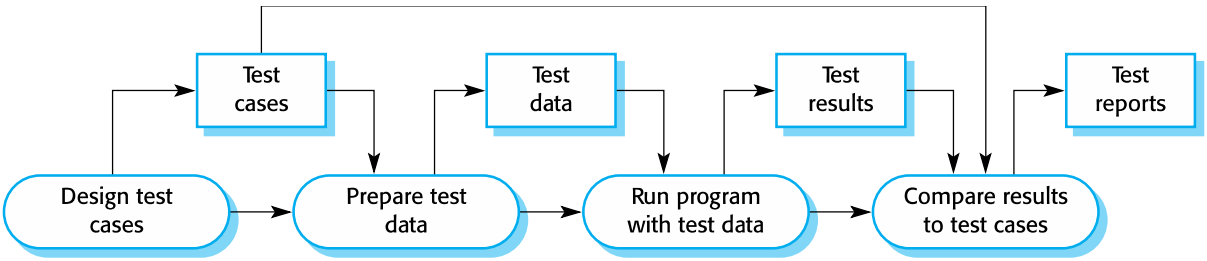
Software inspections:

- Aim to discover defects
- System execution is not required for this
- Can be used on any software representation as shown in above diagram

Advantages of inspection:

- Can avoid masking of errors such as correcting one thing may introduce other errors. So, inspection can avoid this
- Additional costs to test certain parts is avoided
- Also considers quality attributes of the program along with defects

Model of software testing process:



- Manual testing and automatic testing difference is that test data is automatically recorded in automatic testing hence when dev team fixes error we can check with same data. Also known as Regression Testing

Stages of Testing:

- Development testing: system testing during development phase to detect bugs. Done by developer or tester
- Release testing: done by QA team separate from dev team before the system is released
- User Testing: where potential users test the system in their own environment such as UAT on beta versions of software

Development Testing:

- Includes all testing activities done by the developing team
- Includes:
    - Unit Testing: individual modules or code written is tested for functionality as soon as it is completed
    - Component Testing: where several modules made are integrated into component and that component is tested
    - System Testing: all the components are integrated and then the whole system is tested (how they communicate)

Unit Testing:

- Involves testing individual components in isolation to check for defects as syntax errors, exceptions etc.
- Units may be functions, classes, components with defined interfaces

Object class testing:

- Testing for classes involves:
    - Checking for class operations
    - Testing its attributes
    - Validating it in all possible states (on, off states etc)
- Inheritance makes it difficult as tests are not localized as parent and child classes are also affected by it

Choosing unit test cases:

- Developed by developer or pair of developer and tester
- Test cases should be checked on input it gives the desired output and should help to detect defects
- Types:

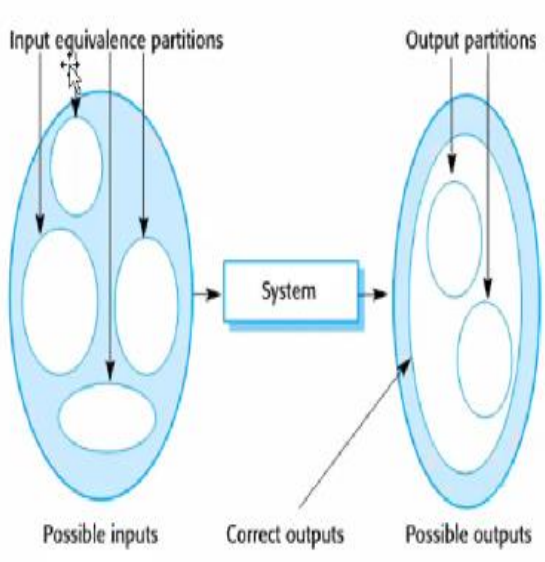
    - Giving galid input should reflect normal flow of the component
    - Providing abnormal inputs should reflect that defect occurs and they are handled properly rather than crashing components

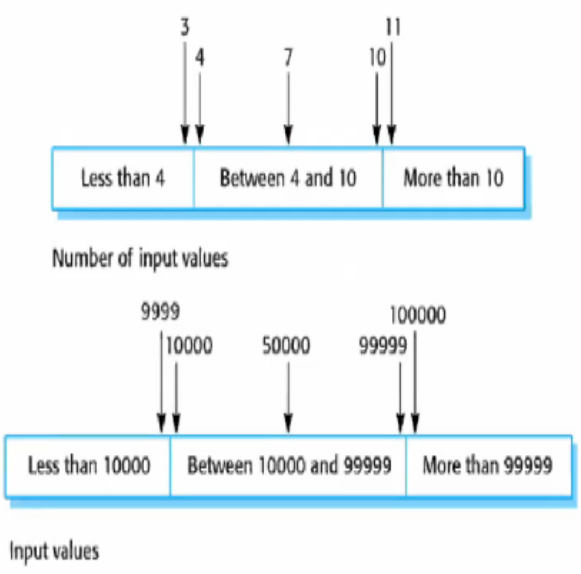
Testing strategies:

- Strategies are made because we don’t have much time to test all factors with all possible cases due to market competition, deadlines etc.
- Partition testing:
    - Group inputs of same characteristics
    - Test-cases from each of these groups is then selected. The result from one test-case should be consistent across all from relevant group
- Guideline testing:
    - Experience developers select and design test-cases based on their intuition and past experiences

Partition strategy:

- Test-cases are made into partitions (groups). When one passes, then the whole partition passes, else fails
- On failing, the test-case should be catered appropriately by the program instead of crashing
- E.g. checking discount on an online shoppnig mart is by provided test input from categories as 1-40, 40-100, 100-200 etc. to check whether discounts are provided in appropriate ranges or not etc.
- Blue are represented are the exceptions





General testing guidelines:

- Guidelines by the book author:
    - Choose inputs to generate all error messages
    - Choose input that cause buffer overflow
    - Repeat same input sets numerous times
    - Force invalid outputs
    - Force outputs that are too small or too large

Component Testing:

System Testing:

- Software is a generic term for any computer code. Its just a solution to a problem which is a useless component on its own
- System is a software that provides services to other software. Its a complete package including the software, dependencies, platform, hardware and other components needed to run the software.
- System testing refers to integrating the components into a system and then testing the system i.e interactions, communication between components in that system.
- This also helps to check compatibility across software components

System and component Testing:

- Integration into a system may involve reusable components or integration with off-the-shelf systems too.
- System testing is a collective process unlike unit testing and may involve teams separate from dev team. Might require some help from dev team for system configuration

Use-case testing:

- Use-cases can be used as a based for testing system components and their interactions.
- Sequence diagrams in the use-cases help to model these interactions within the use-cases.

Testing policies:

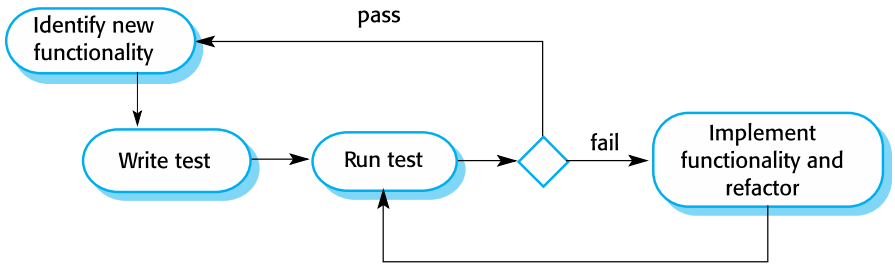
- Exhaustive system testing: test on all possible combinations of the data

- This is time consuming and less effective way to test. Hence, policies must be designed for effective test coverage
- Some examples include:
    - Making partitions of test-cases and validating a single test-cases for a set of cases
    - Testing function sets on user inputs for both correct and incorrect inputs etc.

Test-driven Development:

- This involves making test-cases first and then moving towards the implementation phase (interleaving test and code development)
- Developers have experienced that before implementing they have a mindset of the possible inputs and their respective output and design functionality accordingly
- Code and testing are done in increments. Don't move to the other increment until the code is tested properly.
- Compatible with agile and XP. However, can also be used with plan-driven approaches

TDD process activities:



Benefits of TDD:

- Code Coverage: we have a relevant test for each code snippet written hence, fully tested code
- Regression Testing: After each modification the code is passed through same test cases, until the desired functionality is achieved
- Simplified Debugging: easy to identify in which part the problem occurred due to test for smaller set of codes
- System Documentation: Tests are documented hence documentation is detailed and crystal clear.

Regression Testing:

- Testing the system to check that modifications have not disrupted the code. Tests must be passed before committing code
- With manual testing its expensive, however automatic testing makes it simple as tests are automatically recorded.
- Regression Testing is an important strategy to reduce the side-effect of major changes

Path Testing:

- Conditional statements can lead to multiple execution paths in code.
- Hence, path testing involves ensuring that test-cases must be executed for each execution path atleast once
- Conditional statements are nodes in the flow graph
- Steps:
    - Draw control graph
    - Find set of paths
    - Generate test-cases to exercise each path
- Once path is generated, then we will make test-cases (set of inputs) on which the test-case passes and the execution follows the path specified
- Dynamic program analyzer: a tool to specify and test all the execution paths in your program

Release Testing:

- The testing done before a system release it deployed or handed over to the customer is known as release testing. It comes after system integration and testing
- It is done outside the development team, done by QA staff, hence known as black-box testing
- Non-functional requirements are also tested by release testing. It is done to ensure supplier that software is good for use

Release testing and system testing:

- Similarities:
    - Component integration is done and the whole system is tested in both testing phases.
- Differences:

    - Different teams are deployed for testing each. Dev team checks for bugs by system testing (defect testing) while the QA team check whether software is conform with the specification and requirements (validation testing)

Requirement based Testing:

- Involves making test-cases for each requirement listed by the customer
- Example:

Requirement	Tests
Patient allergic to any medicine then its prescription should result in warning message	- Set up record with no allergies. No warning should be issued - Set record with known allergy. Warning should be issued - Set record with allergy to more than 1 drug. Warning relative to each drug should be issued
Prescriber ignores any warning then must list reason for ignoring	- Set record with allergy and ovverrule the warning if generated. System must require user to input reason for overrule

- Extract features from user story and then provide relevant test cases for each feature

Performance Testing:

- Done to test only non-functional requirements such as performance and reliability. Checks how many resources consumed when an operation is carried by system. Also check speed of system on bearable load
- Performance test involve gradually increasing load on system until performance becomes unacceptable
- We also set benchmarks that on certain load percentages, what will be the system behaviour
- Subsets are stress and load testing.

Stress Testing:

- Deliberately overloading system in order to test its failure behaviour

- Aim is to identify breaking point of system

Load Testing:

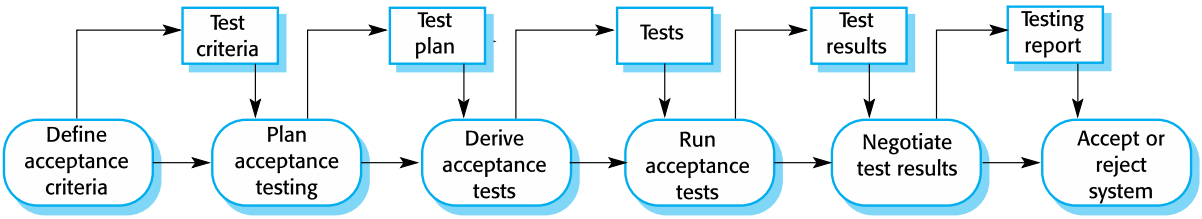
- Testing maximum load in terms of software accessing and large manipulation of data.
- Can be used at both normal and peak conditions and assess relevant behaviour

LOAD TESTING	STRESS TESTING
Load Testing is performed to test the performance of the system or software application under extreme load.	Stress Testing is performed to test the robustness of the system or software application under extreme load.
In load testing load limit is the threshold of a break.	In stress testing load limit is above the threshold of a break.
In load testing, the performance of the software is tested under multiple number of users.	In stress testing, the performance is tested under varying data amounts.
Huge number of users.	Too much users and too much data.
Load testing is performed to find out the upper limit of the system or application.	Stress testing is performed to find the behavior of the system under pressure.
The factor tested during load testing is *performance*.	The factor tested during stress testing is *robustness* and *stability*.
Load testing determines the operating capacity of a system or application.	Stress testing ensures the system security.

User Testing:

- Testing is done in order to gain feedback from customer.
- Done after release testing
- Exhaustive testing cannot be done, hence user is brought with their own mental model to test the system
- Types:
    - Alpha Testing:
        - Agile approach
        - User is with the dev team during testing at developer's site
    - Beta Testing:
        - Made for generic software
        - Users run software in their own environments and give feedback based on it
    - Acceptance Testing: (UAT)
        - Primary for customer systems
        - Customers are brought in and they test the software to check whether it is ready to be deployed or not.
- Acceptance and beta are different as in beta testing, user cannot directly communicate with the development team and their actions are logged and given to the dev team while in case of acceptance testing, customers can give direct feedback to dev team.

Acceptance testing process:



- Ovals represent the stages

Agile methods and acceptance testing:

- Same as alpha testing hence UAT is not done seperately
- Tests defined by user are integrated with other test-cases by the dev team, to be run on the system
- However, problem is that does the 'user' cover all the interests of all system stakeholders?