SECTION two

2. Testing throughout the software life cycle

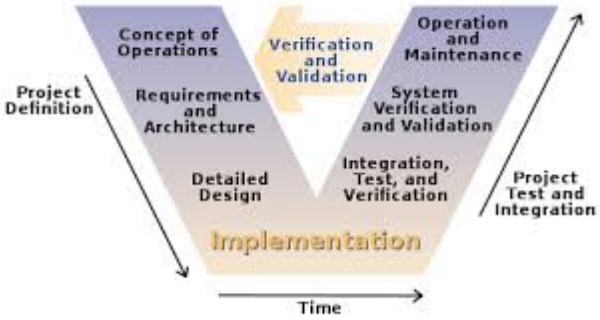
V-Model

- V-model is a framework to describe the software development lifecycle activities from requirements specification to maintenance. It demonstrates how testing activities(verification and validation) can be integrated into each stage of the lifecycle.

- **Verification:** concerned with evaluation of a work product to determine if it meets the requirements set.

- **Validation:** concerned with evaluation of a work product,component or system to determine whether it meets the user needs and requirements.

- **Test Level:** A group of test activities that are organised and managed together. Examples are component test , integration test, system test and acceptance test.



Four Test Levels in V model

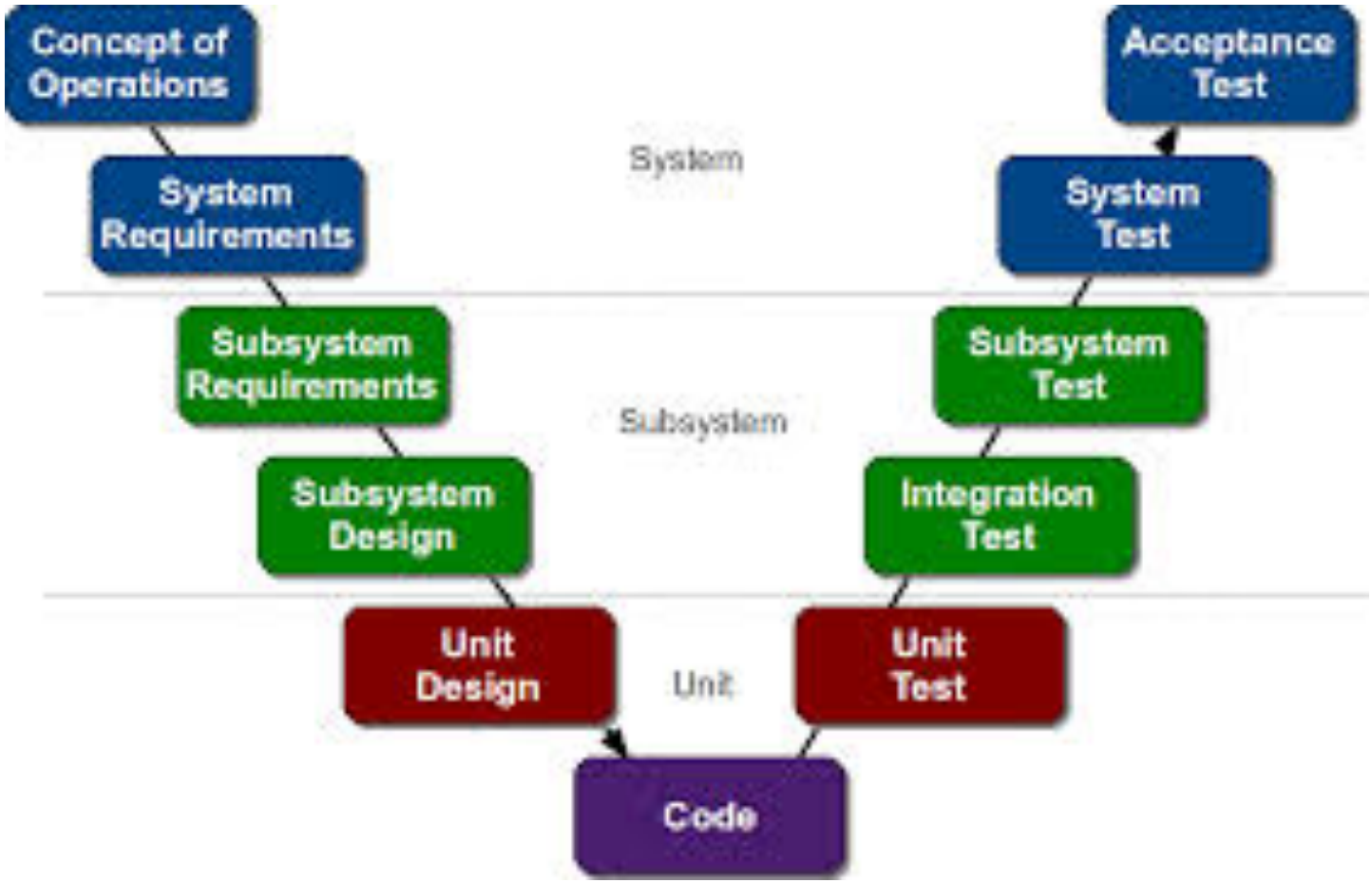
Component Testing: searches for defects in and verifies the functioning of software components(e.g. modules, programs, objects, classes etc..) that are separately testable.

Integration Testing: tests interfaces between components, interactions to different parts of a systems such as an operating system, file system and hardware or interfaces between systems.

System Testing: Concerned with the behaviour of the whole system/project as defined by the scope of a development project or produce. Main focus is verification against specified requirements.

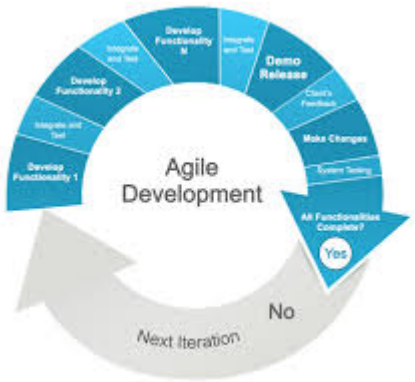
Acceptance Testing: Validation testing with respect to user needs, requirements, and business processes conducted to determine whether or not to accept the system.

V- model


Concept of Operations
System Requirements
Subsystem Requirements
Subsystem Design
Unit Design
Code
Unit Test
Integration Test
Subsystem Test
System Test
Acceptance Test
System
Subsystem
Unit

Testing in an Agile Development Environment

- Testing emphasised in XP and Scrum
- Each iteration culminates an a short period of testing.
- Every time a change implemented – test/s must be ran



Benefits for testers working in an Agile Environment

- Focus on working software and good quality code
- Inclusion of testing as part of and the starting point of software development(test-driven development)
- Accessibility of business stakeholders to help resolve questions about expected behaviour of the system
- Self organizing teams – all responsible for quality
- Simplicity of design – easier to test

Challenges to testers in an Agile environment

- Requirements may be less formal and subject to change
- Developers are doing components testing – Perception testers not required. System testing required even if it doesn't fit into a sprint.
- Less documentation and more personal interaction within an agile team- testers need to adapt and may act more as coaches to both stakeholders and developers
- Less to test in one iteration than a whole system- constant time pressure and less time to think about testing new features.
- Regression testing extremely important with incremental development- taking automated component tests may not make adequate regression suites.


Test Levels

Testing

- Testing critical element of SQA.
- Represents ultimate review of specification, design and coding.
- Often accounts for 40% of total project effort.
- Psychological conflict:
  - SE builds S/W but must now design test cases to demolish the same S/W.
    - **Development - constructive**
    - **Testing - destructive**

Who Tests the Software?

developer

Understands the system

but, will test "gently"

and, is driven by "delivery"

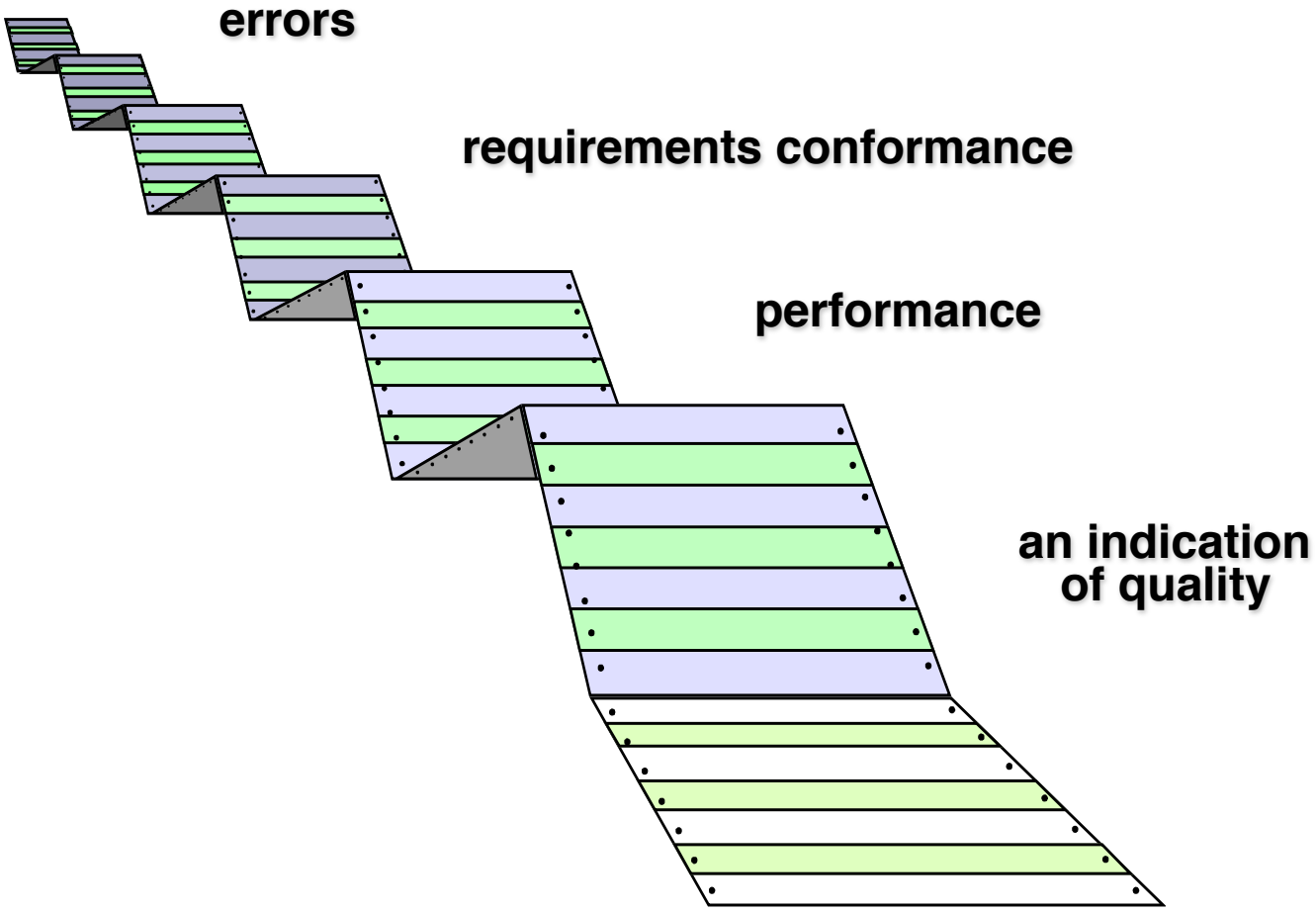
independent tester

Must learn about the system,

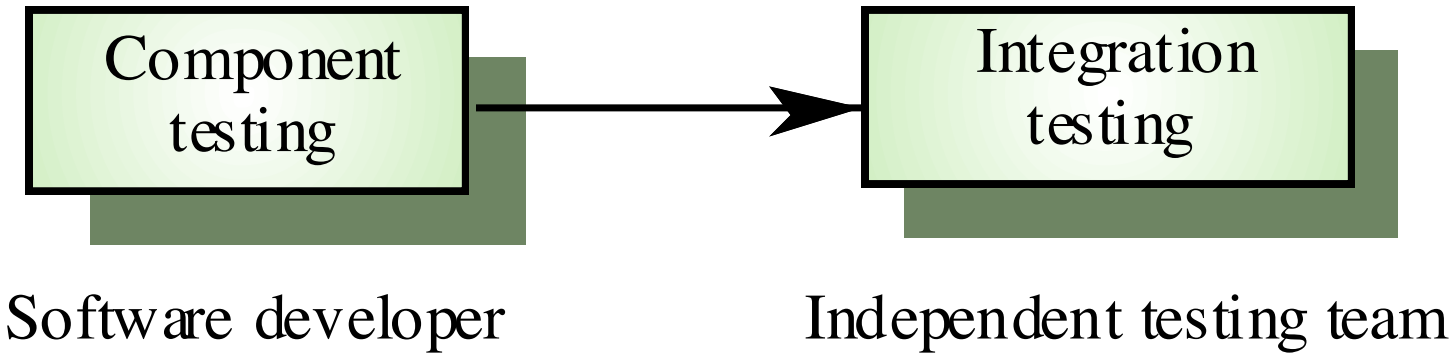
but, will attempt to break it

and, is driven by quality

What Testing Shows



Testing phases



component(Unit Testing)

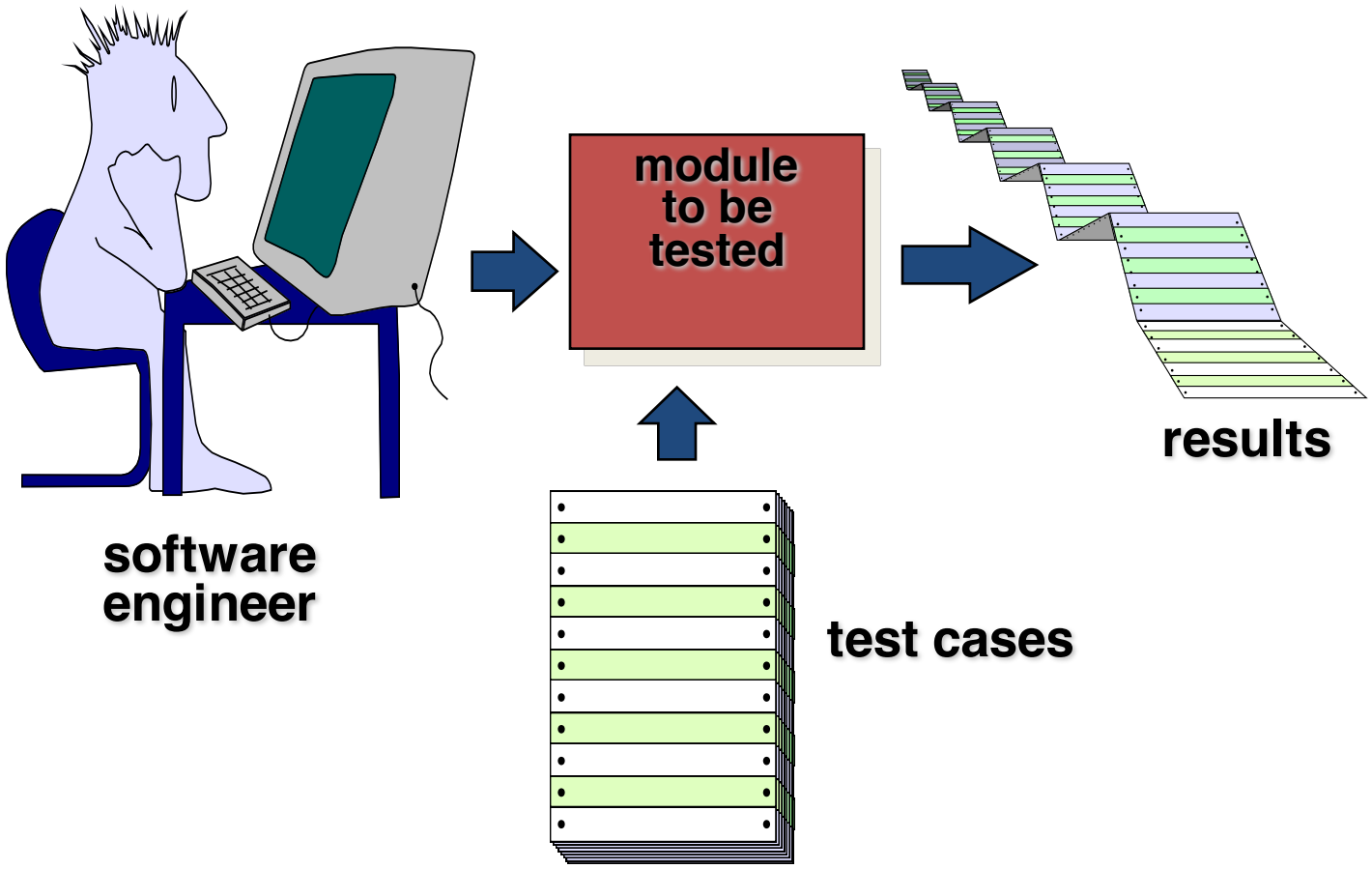
SAMPLE PROJECT

- Our project consists of two modules namely admin and user,

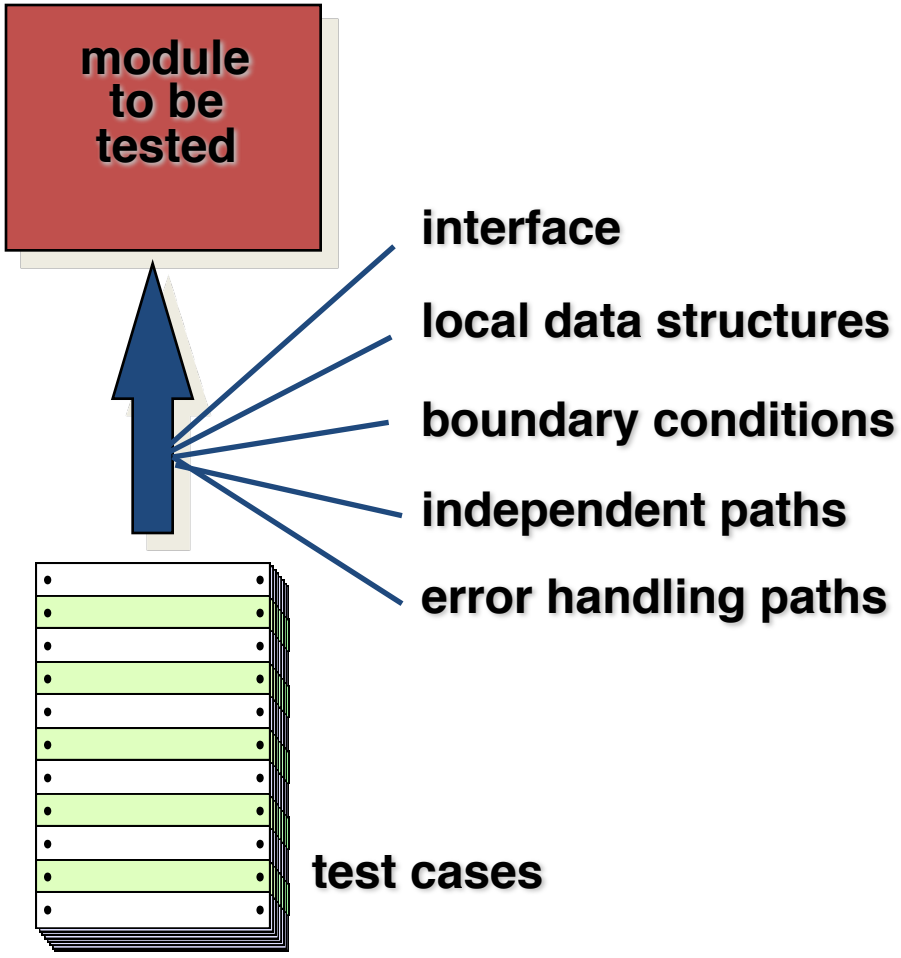
- Admin will do the following functionalities :
  - login,
  - registering the user, a
  - signing new telephone numbers,
  - inserting bills
  - and new plans,
  - viewing the feedback and complaints.
  - and signing out.

User can performs functionalities
  - login,
  - applying for new connections
  - ending the feedback,
  - complaints,
  - viewing the bill
  - new plans
  - and signing out.

Unit Testing(e.g. Junit; TestNG)

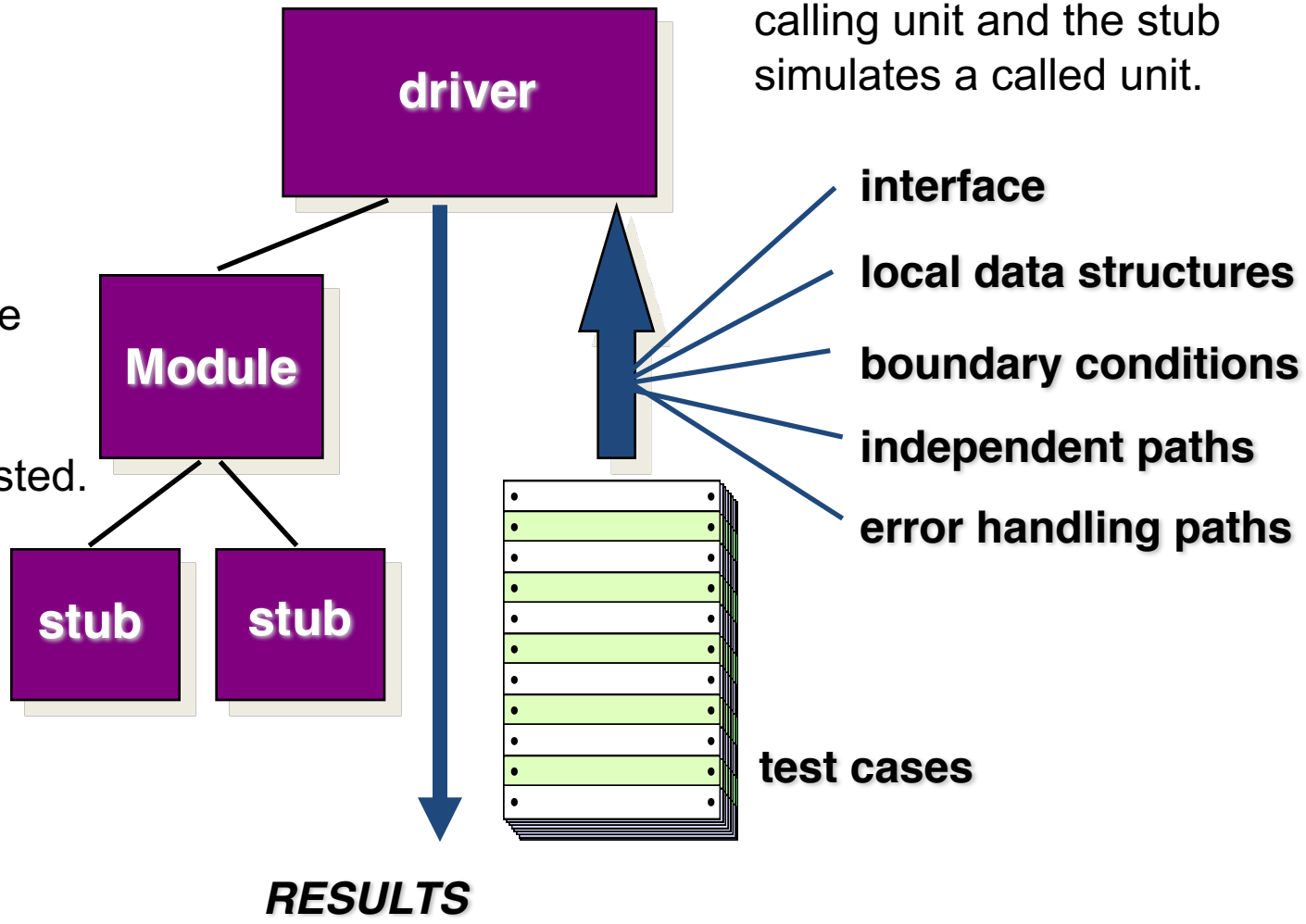


Unit Testing



Unit Test Environment

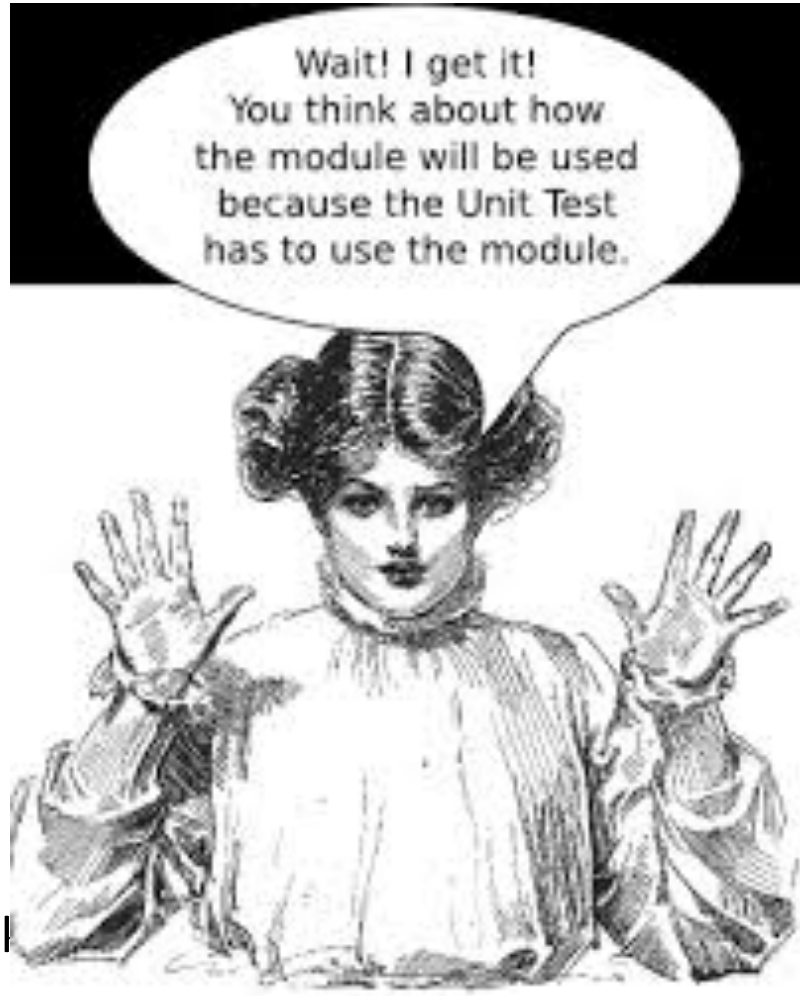
test stubs are programs which simulate the behaviors of software components (or modules) that are the dependent modules of the module being tested.



The driver simulates a calling unit and the stub simulates a called unit.

Unit Testing.

- Deals with the module
- Uses detail design description
- White box oriented.
- Tests the following:
  - module interface
  - local data storage for integrity
  - boundary conditions
  - independent paths
  - error handling paths
  - Non functional characteristics
    - Eg resource behaviour(memory leak



UNIT TESTING OF SAMPLE PROJECT

- **Admin**
- First we start with login module of admin - create application consisting of login functionality .

- Unit testing of login functionality -will check whether the admin after entering the valid login name and password able to enter into his profile or not.

- Assign new telephone numbers, we have two methods namely reset member passwords and delete members . so will do individual testing of this two methods present with in the block user module of admin.

- Once the reset member and delete member application has been created , it will be tested checking the output that is whether the passwords are getting resets or not and whether the members are getting deleting or not , on the invocation of the respective methods of the block user module of Admin.

- Than the logout module will tested – test admin is logging out, he /she is signing out of the application or not.

- **User module:**
- User can performs functionalities such as
  - logging ,
  - sending the messages.

- For the first module , logging application will be created and will be tested whether after entering the valid username and password user is able to enter into his profile or not and also we will test whether after entering the invalid details he is being restricted from entering his profile or not.

- The sending message module consists of two methods namely entering the message which the user wants to send and selecting the appropriate encryption types that are either AES or DES encryption. once after selecting the encryption , the user will send the message.

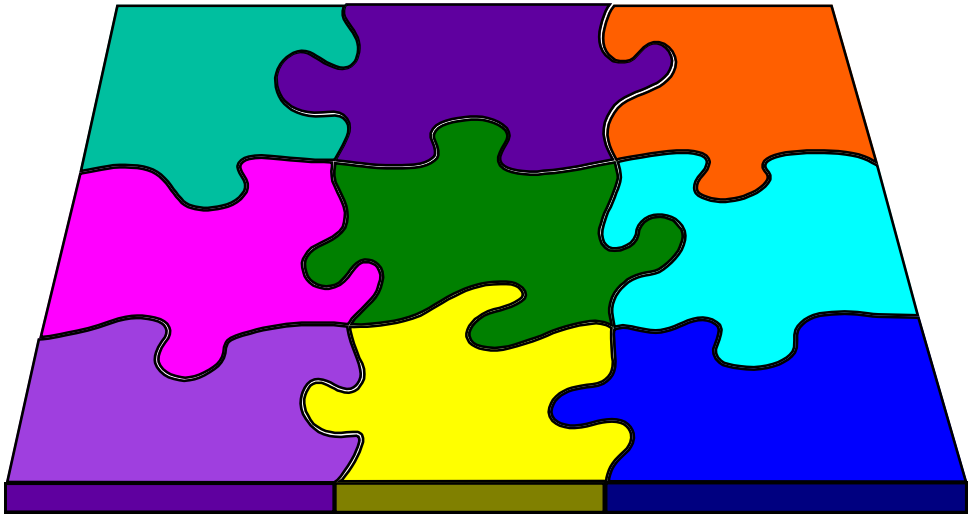
Integration Testing.

- When putting the modules together:
  - data can be lost across the interface.
  - one module can have an adverse affect on another.
- Involves putting the program together and at the same time conducting test to uncover errors at the interface.

Integration Testing Strategies

Options:

- the “big bang” approach
- an incremental construction strategy



Integration Testing.

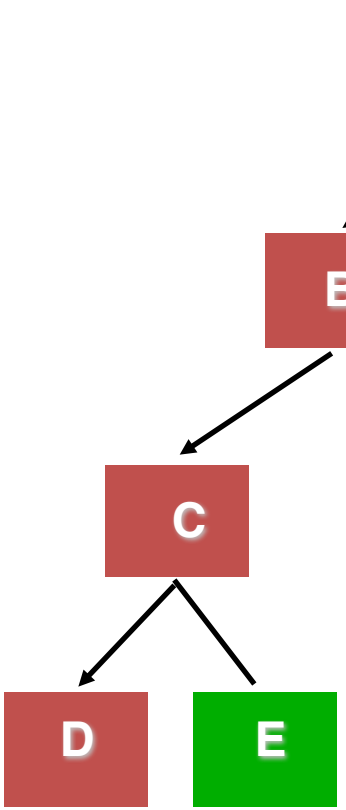
- The ‘big bang’ approach to integration is not very effective and leads to chaos.

- Incremental testing is preferred where the program is put together and tested in segments - errors are easier to isolate and correct.
- Integration can be in a top-down or bottom-up fashion.

Approaches to integration testing

- Top-down testing
  - Start with high-level system and integrate from the top-down replacing individual components by stubs where appropriate
- Bottom-up testing
  - Integrate individual components in levels until the complete system is created
- Functional Incremental: integration and testing takes place on the basis of the functions

Top Down Integration

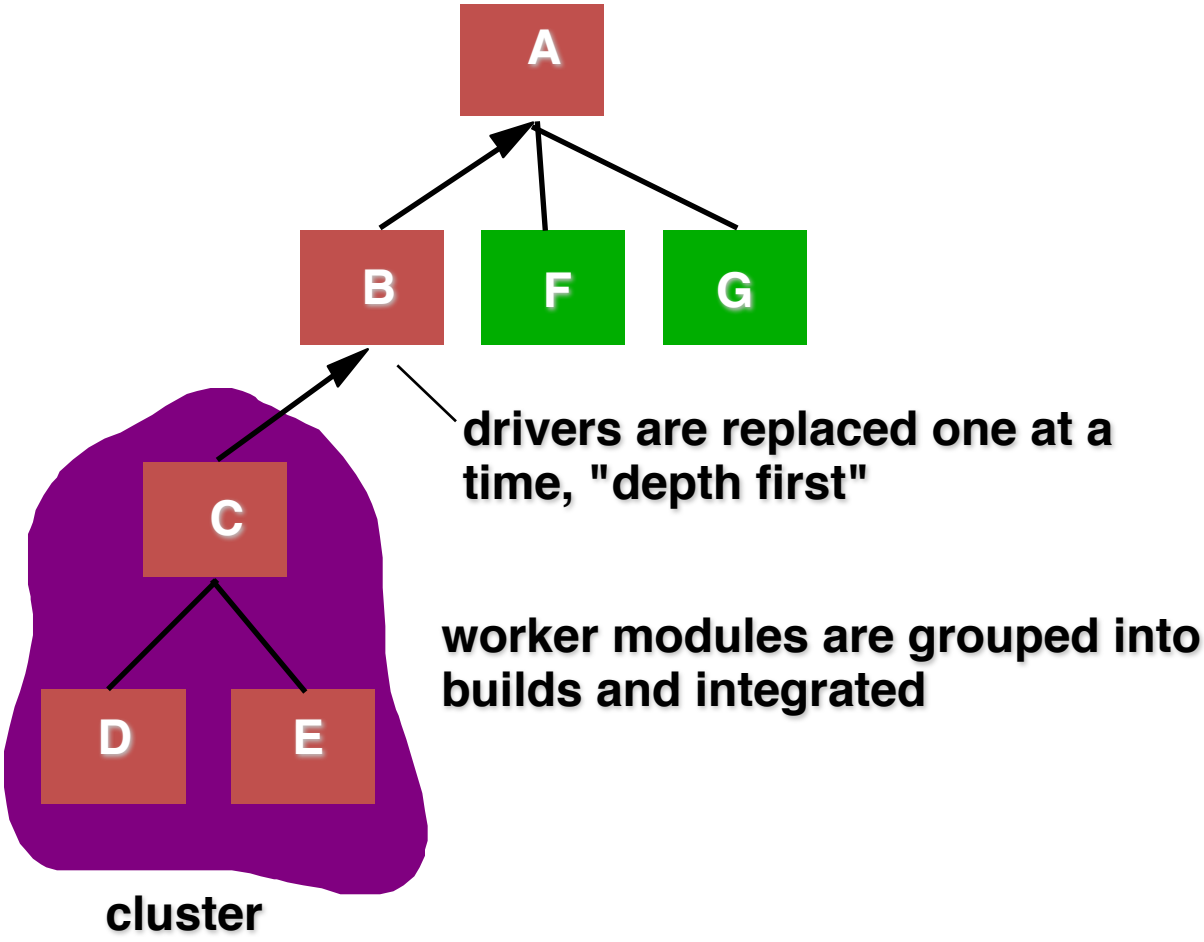


top module is tested with stubs

stubs are replaced one at a time, "depth first"

as new modules are integrated, some subset of tests is re-run

Bottom-Up Integration



Integration TESTING OF SAMPLE PROJECT

Integration testing:

1) Admin module:

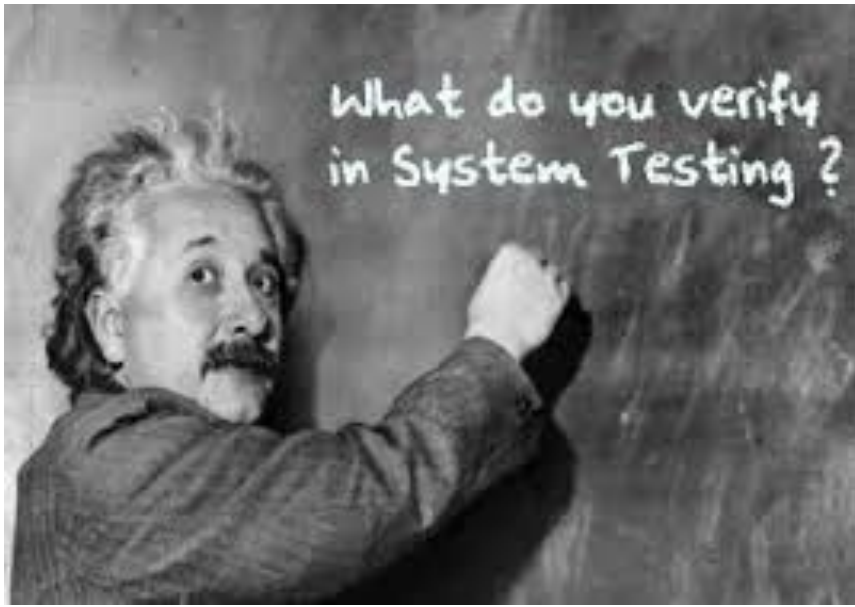
In the integration testing the modules present in the admin module such as logging , registering users, blocking user and signing out. Will be tested on integrating basics that is this functionalities of Admin module will be tested at once and will check whether this module is working perfectly or not

2) User Module:

In this module , all the functionalities of a User module such as logging, sending messages, Inbox and signing out will be tested on a whole and will see whether this module is working perfectly or not ,by including all the validation controls .

System Testing:

- Software only one element of larger system. Idea here is to fully exercise the *entire* system.
- System testing performed to verify it meets specified requirements.
- Interface problems should be anticipated.
- Error handling paths should be designed to test data from other parts of system.



System Testing

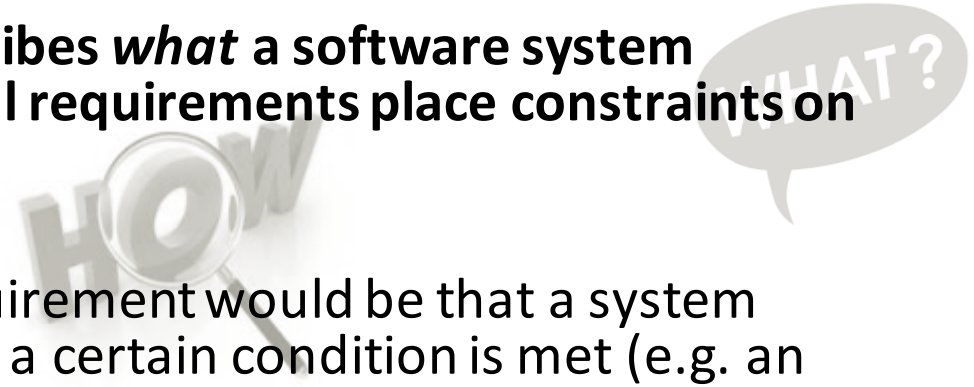
• System Testing most often the final test on behalf of development to verify the system meets the specification

•System testing requires a controlled environment and should correspond to the final target as much as possible in order to minimize the risk of environment specific features not been found during test

•System testing should investigate both functional and non functional requirements

Functional and non-functional requirements

- **A functional requirement describes *what* a software system should do, while non-functional requirements place constraints on *how* the system will do so.**



- An example of a functional requirement would be that a system must send a an email whenever a certain condition is met (e.g. an order is placed, a customer signs up, etc).

- A related non-functional requirement for the system may be that emails should be sent with a latency of no greater than 12 hours from such an activity.
- The functional requirement is **describing the behavior of the system** as it relates to the system's functionality. The non-functional requirement **elaborates a performance characteristic** of the system.

Typically non-functional requirements fall into areas such as:

Accessibility
Capacity, current and forecast
Compliance
Documentation
Disaster recovery
Efficiency
Effectiveness
Extensibility
Fault tolerance
Interoperability
Maintainability
Privacy
Portability
Quality
Reliability
Resilience
Response time

Test Levels

Acceptance Testing

- Formal testing with respect to user needs, requirements and business processes

- Acceptance test answers questions such as:
  - Can the system be released?
  - What if any are the outstanding business risks?
  - Has development met their obligations?

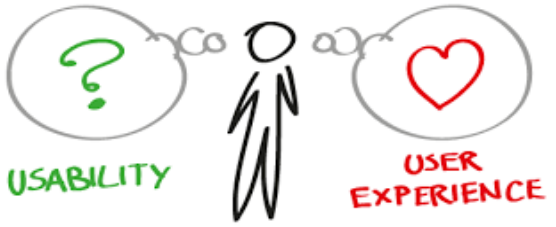
Acceptance Testing

- Acceptance testing is most often focused on a **validation** type of testing – determine if the system is fit for purpose
- Establish confidence in the system eg usability of the system
- Finding defects is not the main purpose
- **Acceptance testing may occur at more than just a single level**
  - A COTS (Commercial off the shelf) software product may be acceptance tested when it is installed or integrated
  - Acceptance testing of the usability of a component may be done during component testing
  - Acceptance testing of a new functional enhancement may come before system testing

Acceptance Testing

Two main test types:

1. **User Acceptance Test** – focus on functionality validating fitness for purpose



- Performed by users and application managers

- Links tightly to system test and may overlap in time
  - E.g performance - load time may be an issue, usability of product.

2. Operational Acceptance Test(Production Acceptance Test) – validates if system meets requirements for operation.

- System administration will perform these tests before system released
  - E,g Testing of backup/restore, data load and migration tasks, disaster recovery, user management, maintenance tasks, and periodic check of security vulnerabilities

Other types of Acceptance Testing

- **Contract Acceptance Testing**: Performed against a contract's acceptance criteria for producing custom-developed software

- **Regulatory Acceptance Testing**: Performed against the regulations eg governmental, legal or safety regulations.

  - The first is Alpha Testing, this takes place at the developers site, a cross-section of potential users are invited to use the system, and developers
  - The second is Beta Testing,.

Alpha and Beta Testing

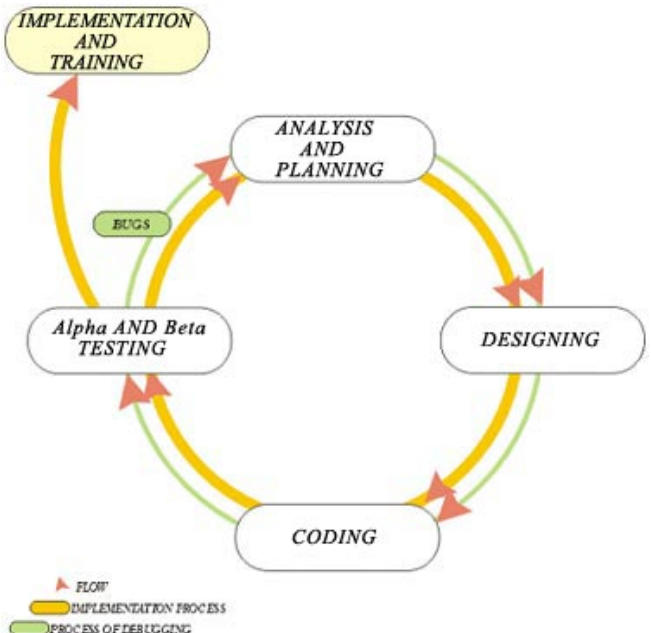
- **Alpha and Beta testing**:
- Developer cant foresee problems that might arise when the user gets hands on the software.
- Acceptance tests are conducted when building custom software for just one customer. Time allows the customer to ‘test drive’ the software over an extended period.

Alpha and Beta Testing

- If the software is intended for the mass market, customer testing is impractical, but feedback is needed, so it's often done in a two stage process
- A and B tests are performed where software for wide distribution.
- Alpha test:
  - conducted at developers site by the customer supervised by the developer who observe the users and note problems.
  - conducted in a controlled environment.

Alpha and Beta Testing

- Beta test:
  - conducted by cross-section of the users (live application of the s/w), who install it, and use it under real-world conditions. The users send records of incidents with the system to the development organisation where the defects are repaired
  - Developer not present.
  - User reports recorded results and problems to the developer on a regular basis.



Test Types

FUNCTIONAL TESTING

- **Test Type:** Focuses on a particular test objective – could be a function of the system, a non functional requirement, looking for unintended changes, confirming that defects have been fixed.

- **Function of a system is what it does.**

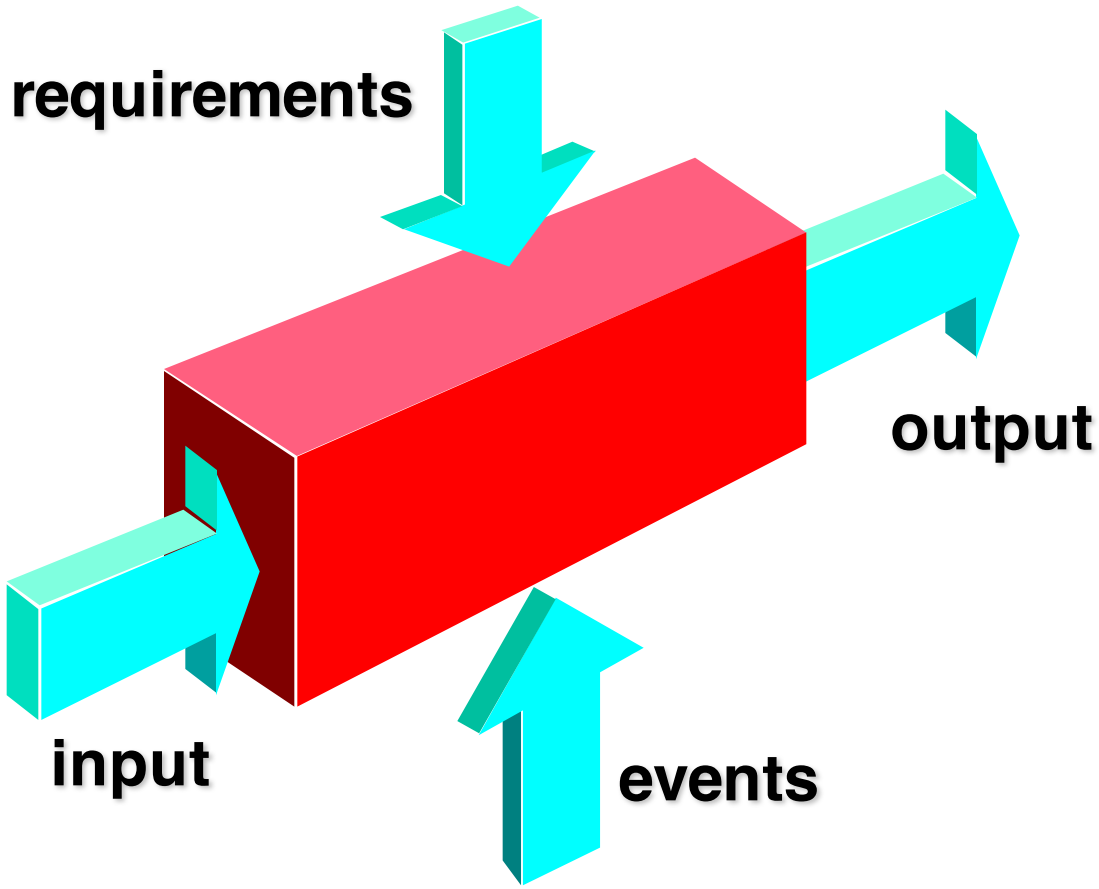
- **Functional Testing** considers the specified behaviour,(Often referred to as Black Box Testing- inaccurate **as black box testing** includes non functional requirements). Specification based.

- Done from two perspectives

1. Requirements-based testing: Take table of contents of requirements specification, prioritizing on risk criteria, Most critical tests will be included

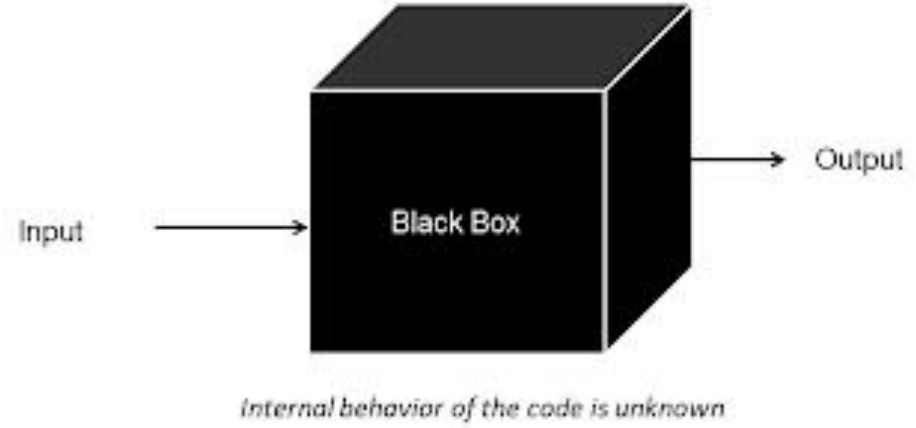
2. Business process based testing: uses knowledge of business processes. Scenarios and use cases – day to day business use of the system
e.g Payroll system – someone joins a company, they are paid monthly, if they leave the company must be removed from payroll

Black-Box Testing

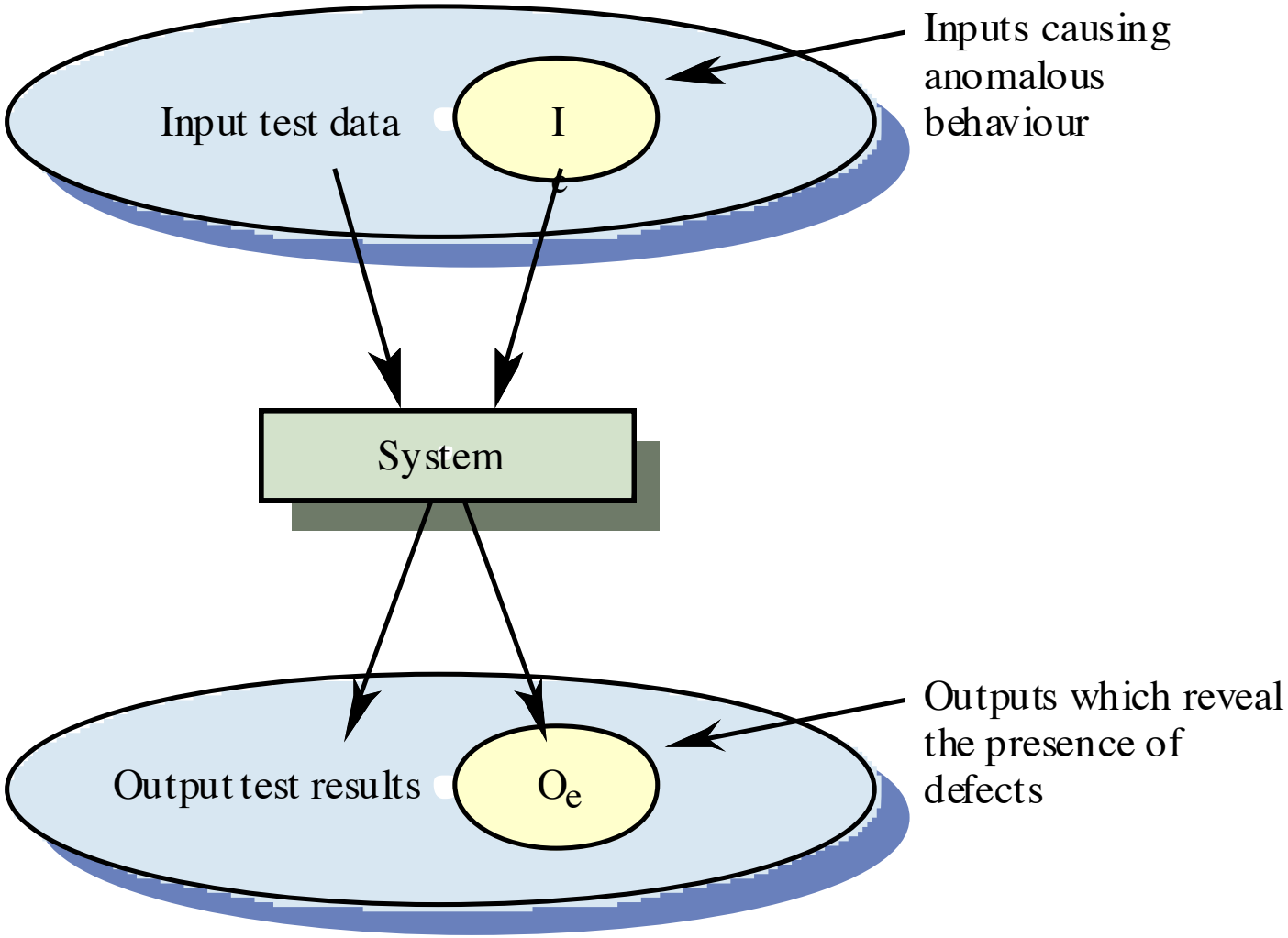


Black Box Testing

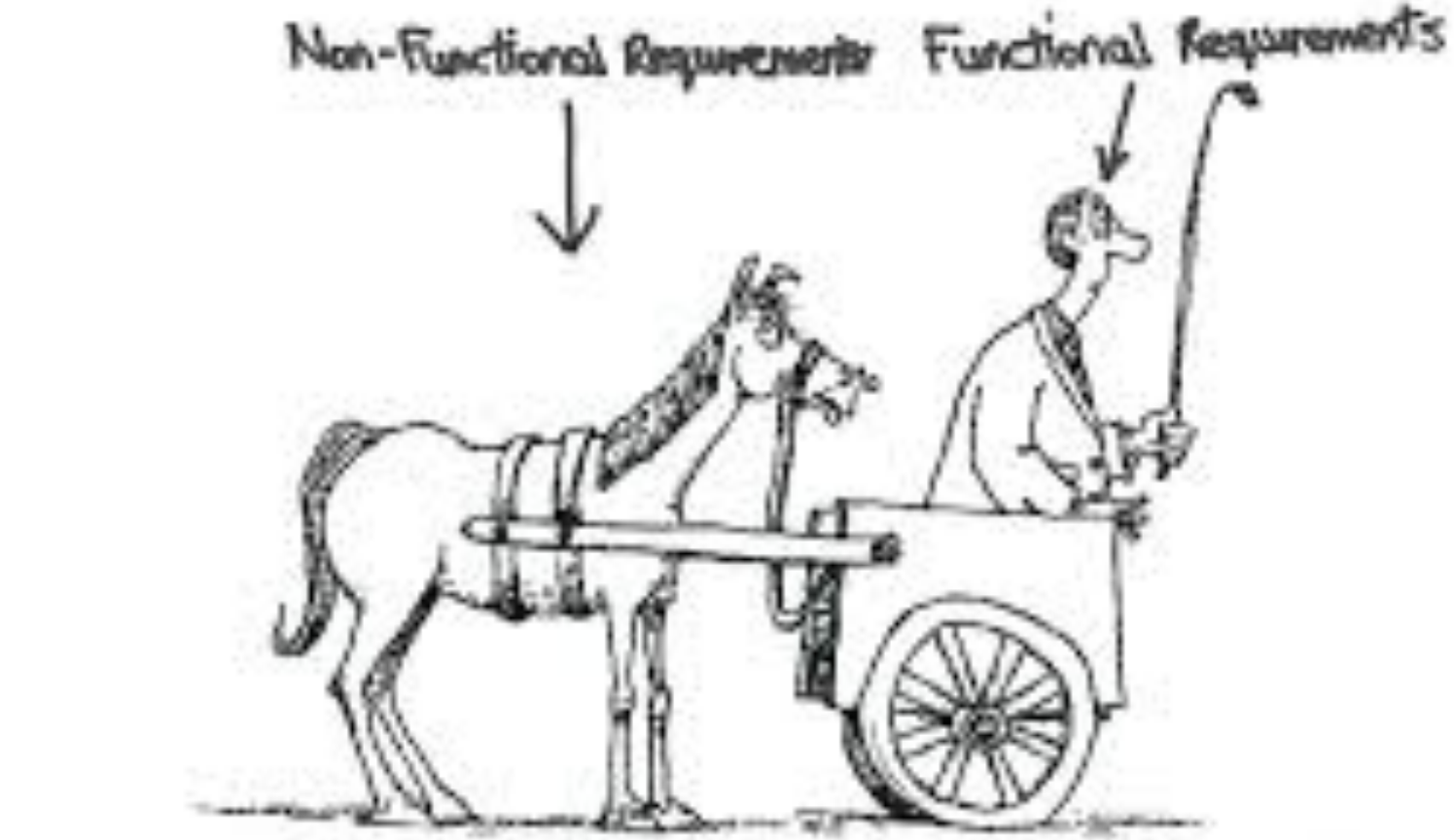
- Demonstrates that:
  - functions are operational
  - input accepted properly
  - output correctly produced
  - integrity of external information (e.g. files) is maintained
- Black Box testing occurs at the interface and takes little account of the internal logic.



Black-box testing



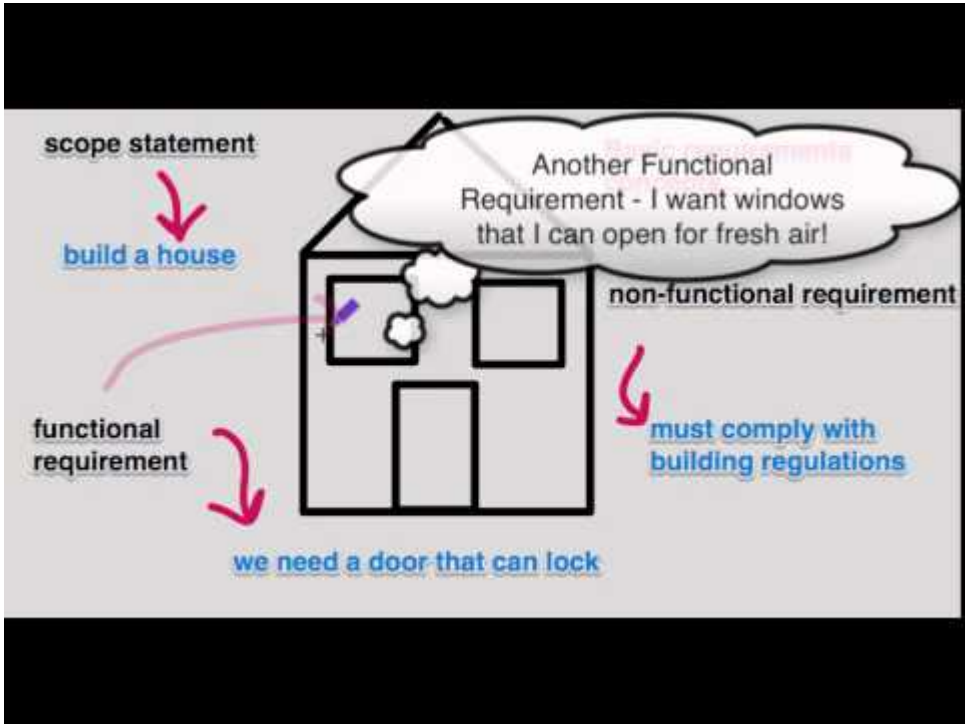
NON FUNCTIONAL TESTING



- **Non functional Testing** :Testing of the quality characteristics- **how well or how fast something is done- how well the system works**

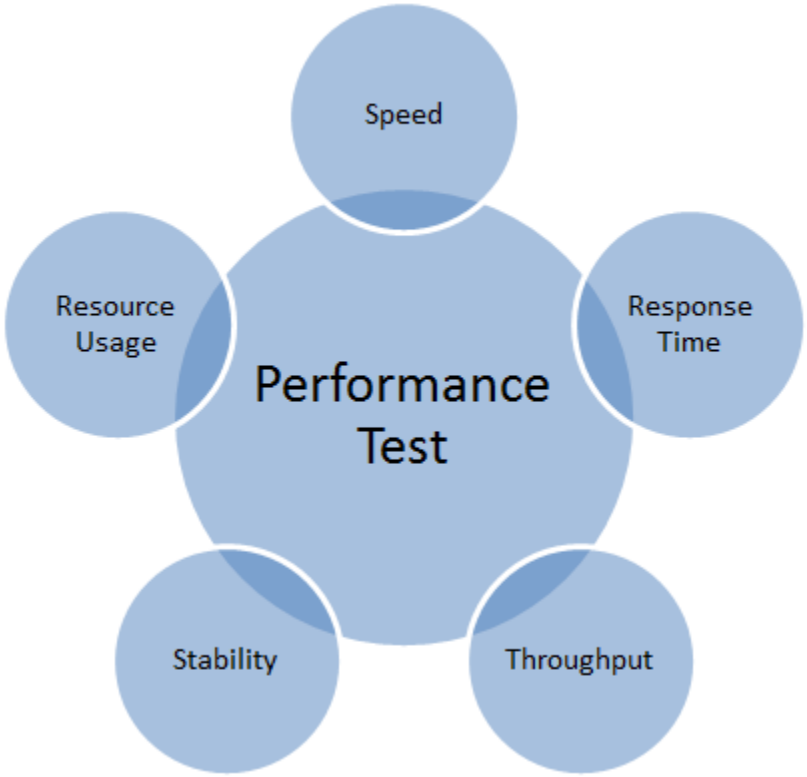
- Defines expected results in terms of external behaviour of software - typically black box testing

- Non functional testing performed at all test levels and includes the following but not limited to the following tests



Performance Testing

Testing to determine the performance of a software product
eg resource utilisation and response times



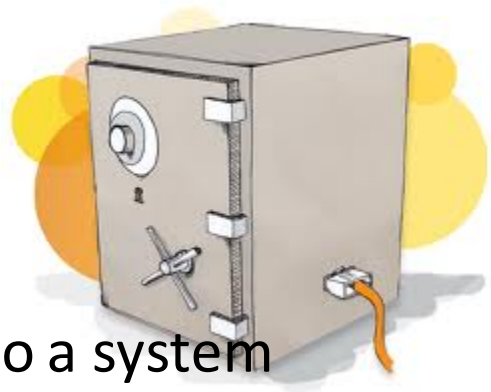
Usability testing

Testing for 'user-friendliness'. Clearly this is subjective, and will depend on the targeted end-user or customer. User interviews, surveys, video recording of user sessions, and other techniques can be used.

Stress testing

- Exercises the system beyond its maximum design load. Stressing the system often causes defects to come to light
- Stressing the system tests failure behaviour.. Systems should not fail catastrophically. Stress testing checks for unacceptable loss of service or data
- Particularly relevant to distributed systems which can exhibit severe degradation as a network becomes overloaded

High Order Testing



Security testing

- verifies that protection mechanisms built into a system will, in fact, protect it from improper penetration

Load testing - testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.

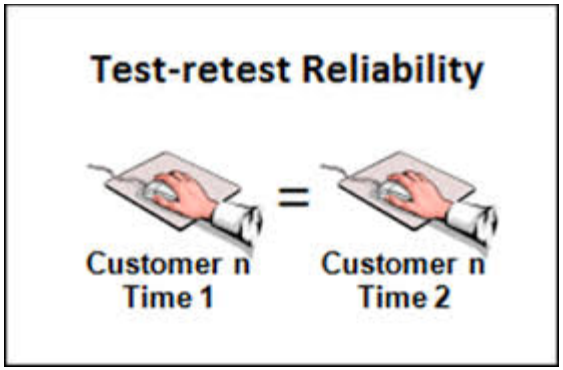
Types of Tests

- Example of load test:
- if a web site with a shopping cart is intended for 100 concurrent users who are doing the following functions:

- 25 Virtual Users are browsing through the items and logging off
- 25 Virtual Users are adding items to the shopping cart and checking out and logging off
- 25 Virtual Users are returning items previously purchased and logging off
- 25 Virtual Users are just logged in without any activity
- Using various tools available to generate these VUsers, the application is subjected to a 100 VUser load as shown above and its performance is monitored.

Maintainability Testing

Testing to determine the maintainability of a software product

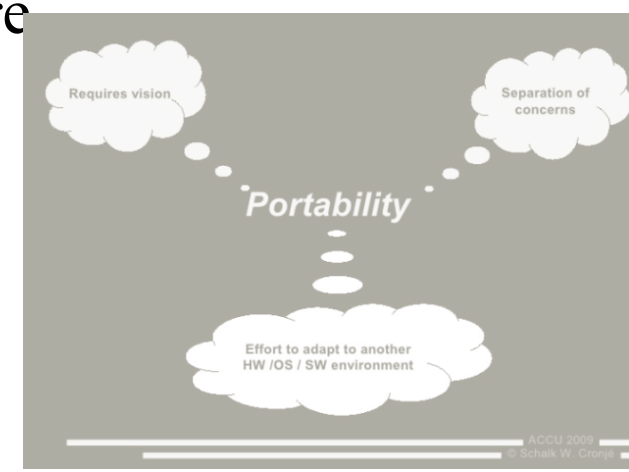


Reliability Testing

Testing to determine the reliability of a software product

Portability Testing

Testing to determine the portability of a software product



ISO 9126 – Six Quality characteristics for software

- **Quality Model**
- **Functionality** - *The functions are those that satisfy stated needs.*
  - Suitability
  - Accuracy
  - Interoperability
  - Compliance
  - Security

- **Reliability** - *Capability of software to maintain its level of performance*
  - Maturity
  - Recoverability
  - Fault Tolerance

ISO 9126

- **Usability** –
  - Learnability
  - Understandability
  - Operability

- **Efficiency** - *level of performance of the software and the amount of resources used*
  - Time Behaviour
  - Resource Behaviour

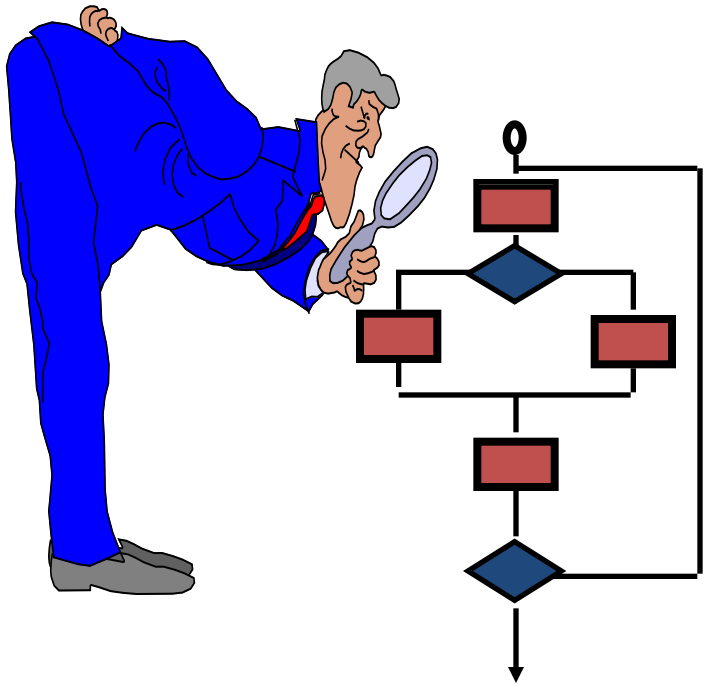
- **Maintainability** -.
  - Stability
  - Changeability
  - Testability

ISO 9126

- **Portability** - *software to be transferred from one environment to another.*
  - Installability
  - Replaceability
  - Adaptability

- The standard provides a framework for organizations to define a quality model for a software product. On doing so, however, it leaves up to each organization the task of specifying precisely its own model. This may be done, for example, by specifying target values for quality metrics which evaluates the degree of presence of quality attributes.

White-Box Testing(Structural Testing



... our goal is to ensure that all statements and conditions have been executed at least once ...

White Box Testing

It requires programming skills to identify all paths through the software.

- In electrical hardware testing, every node in a circuit may be probed and measured; an example is in-circuit testing.

- Since the tests are based on the actual implementation, if the implementation changes, the tests probably will need to change, too. For example ICT needs updates if component values change, and needs modified/new fixture if the circuit changes.

Maintenance Testing

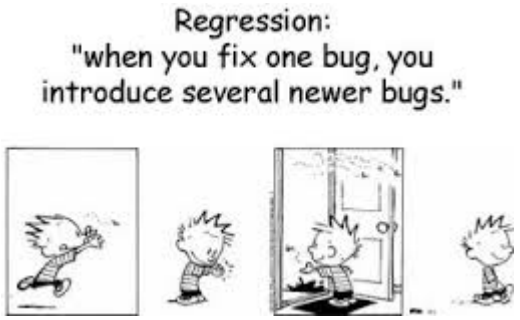
- Once a system is deployed it is in service for years and decades. During this time the system and its operational environment is often corrected, changed or extended. Testing that is provided during this phase is called maintenance testing.

- Modifications can result from
  - Minor Releases – Bug Fixes and New Features
  - Corrective and more urgent emergence changes
  - Changes of environment – operating system, database upgrades, patches to vulnerabilities in operating system
  - Migration (moving from one platform to another) – must test new environment
  - Data from one application migrated into system been maintained

Usually maintenance testing is consisting of two parts:

- **First** one is, testing the changes that has been made because of the correction in the system or if the system is extended or because of some additional features added to it.

- **Second** one is regression tests to prove that the rest of the system has not been affected by the maintenance work.



- An **impact analysis** – the assessment of change to the layers of development, test documentation and components in order to implement a given change to specified requirements.

- Decision made with stakeholders as what parts of system unintentionally affected and may need more regression testing
- Ad-hoc corrective modifications – like first aid and standard test process followed at a later stage(network failing with users on line, production run fails late at night). Risk analysis should be performed to establish which programs/ functins constitute greatest risk.