SOFTWARE ENGINEERING

PRABHJOT KAUR
CSE DEPARTMENT

REQUIREMENT ENGINEERING

- **Requirement:- is a feature or description of system.**

Requirements describe

What not How

- **Requirement Engg:- Understanding the requirements of customer + Documenting the Requirements.**

- **WORK DONE:- RE produces one large document written in natural language, containing a description of WHAT s/m will do without describing HOW it will do it. (SRS)**

- **INPUT to RE:- Problem statement prepared by customer i.e overview of existing s/m + new or addn. Functionality to be added.**

REQUIREMENT ENGINEERING PROCESS

It is a four step process, which includes –

1. **Requirement Gathering(ELICITATION):-Requirements are identified with the help of customer.**
   - **focused towards goal of the organization.**
   - **It comes up with rough idea about what all functions the software must perform and which all features are expected from the software.**

2. **Requirement Analysis:- It analysed to find inconsistencies, contradictions, priorities AND omit some reqmts.**
   - **how the intended software will interact with hardware**
   - **speed of operation**

(CONTD.)

3 . Software Requirement Specification (DOCUMENTATION):- Leads to the preparation of SRS.

- SRS is a document created by system analyst after the requirements are collected from various stakeholders.
- User Requirements are expressed in natural language.
- Design description should be written in Pseudo code.
- Conditional and mathematical notations for DFDs.

4. Software Requirement Validation:- To improve the quality of SRS.

- If they are valid and as per functionality and domain of software.
- If they are complete

SOFTWARE REQUIREMENTS ELICITATION

- RE is an activity that helps us to understand what problem has to be solved & what customer expect from the sw.
- In this the effective communication b/w customer & developer is important.
- Not an easy method as many hurdles occur in it.

- METHOD:- Developers ask questns to customers-> responds with ans-> then cross questioning.

- HURDLES:- Misunderstanding, Communication Gap.

- REASONS for conflict b/w concerned people:-

  ??????

REQUIREMENTS ELICITATION METHODS

There are a number of requirements elicitation methods.

- Interviews
- Brainstorming Sessions
- Facilitated Application Specification Technique (FAST)
- Quality Function Deployment (QFD)
- Use Case Approach

1. INTERVIEWS

- Main purpose is to understand each other or understand customer's expectations from the software.
- Requirement Engineer act as a mediator b/w customer and development team.
- So correct approach is:- open-minded, co-operative, understanding, expertise.

TYPES OF INTERVIEWS:-

- In open ended interviews, there is no pre-set agenda. No pre-defined questions is prepared.
- In structured interview, agenda of fairly open questions is prepared. Sometimes a proper questionnaire is designed for the interview.
- Questions should be clear, short & simple

2. BRAINSTORMING SESSIONS

- Group discussion.
- Highly trained facilitator required.
- Every idea is documented so that everyone can see it.
- Platform to express & share your views & expectations & difficulties in implementation.

- **WORK PRODUCT**:- Document is ready
  - Ideas are documented to be visible to each participant.
  - Detailed report, containing each idea in simple lang., is prepared & reviewed by facilitator.
  - Finally a document is prepared which consists of the list of requirements and their priority if possible.

3. FACILITATED APPLICATION SPECIFICATION TECHNIQUE

- **Bridge the expectation gap – difference between what the developers think they are supposed to build and what customers think they are going to get.**

- **A team oriented approach is developed.**

4. QUALITY FUNCTION DEPLOYMENT

- customer satisfaction is of prime concern.

- 3 types of requirements are identified –
- Normal requirements:– proposed software are discussed with the customer.

- Expected requirements:– These requirements are so obvious that the customer need not explicitly state.

- Exciting requirements:– Example – when an unauthorised access is detected, it should backup and shutdown all processes.

5. USE CASE APPROACH

- **Combines text and pictures.**
- **The use cases describe the ‘what’, of a system and not ‘how’.**
- **components of the use case deign includes three major things – Actor, Use cases, use case diagram.**

a) Actor – External agent. An actor maybe a person, machine.

Actors can be primary actors or secondary actors.

(CONT.)

b) Use cases – They describe the sequence of interactions between actors and the system. They capture who do what with the system.

c) Use case diagram – Graphically represents what happens when an actor interacts with a system.

A stick figure is used to represent an actor.

- An oval is used to represent a use case.
- A line is used to represent a relationship between an actor and a use case


sing up
Verification
Forgot Password
Login
Manage Account
Add Interests/ Passion
view profiles od people with similar interest/Passions
Follow someone
User
System
chat with other user
Post Photots/ Videos/Recording
share post of other user on own acc
Like/comment on ony post
categorize the posts according to the category selected by user
post mystery under a special category
Solve mystery cases
Promote posts of users who have solved mysteries faster than others or who have solved more number of mysteries.
User
System

SOFTWARE REQUIREMENTS

- **Functional Requirements:-** **(Product features)**
- **Describe what the s/m has to do**
- **What are the expectations from the s/m s/w**
- **What the s/w should not do. (FOR USERS)**
- **define functions and functionality within and from the software system.**

- **EXAMPLES -**

- **User should be able to mail any report to management.**

- **Users can be divided into groups and groups can be given separate rights.**

(CONT.)

- **Non Functional requirements:- are mostly quality requirements.**
- **Stipulate how well the software does, what it has to do.**
- **Eg:- FOR USERS :- Performance, Reliability, Usability.**
  - **For DEVELOPERS:- Maintainability, Portability, Testability after delivery of SW.**

- **User requirements:- written for users who are not experts of sw field.**
- **Highlight the overview of s/m without design description.**
- **Specifies:- quality, constraints, external behaviour.**

(CONT.)

- **System requirement:- derived from user requirements or an expanded form of user reqmnts.**
- **Used as input to designers so that they can prepare software design documents.**

- **FEASIBILITY STUDIES:- determines the project is workable or not.**
- **WORK PRODUCRT:- Feasibility report-> it helps the manager, customer, project team to decide if the software should be develop or built.**

BUILDING ANALYSIS MODEL

- Requirement Analysis
- Data modeling Concepts
- Flow Oriented Modeling

This is done in class.

SOFTWARE TESTING APPROACH

- A test approach is the test strategy implementation of a project, defines how testing would be carried out.
- Proactive - An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.
- It is a series of various tests.
- It allows to test, verify and validate the business requirement and application architecture.

STRATEGY OF TESTING

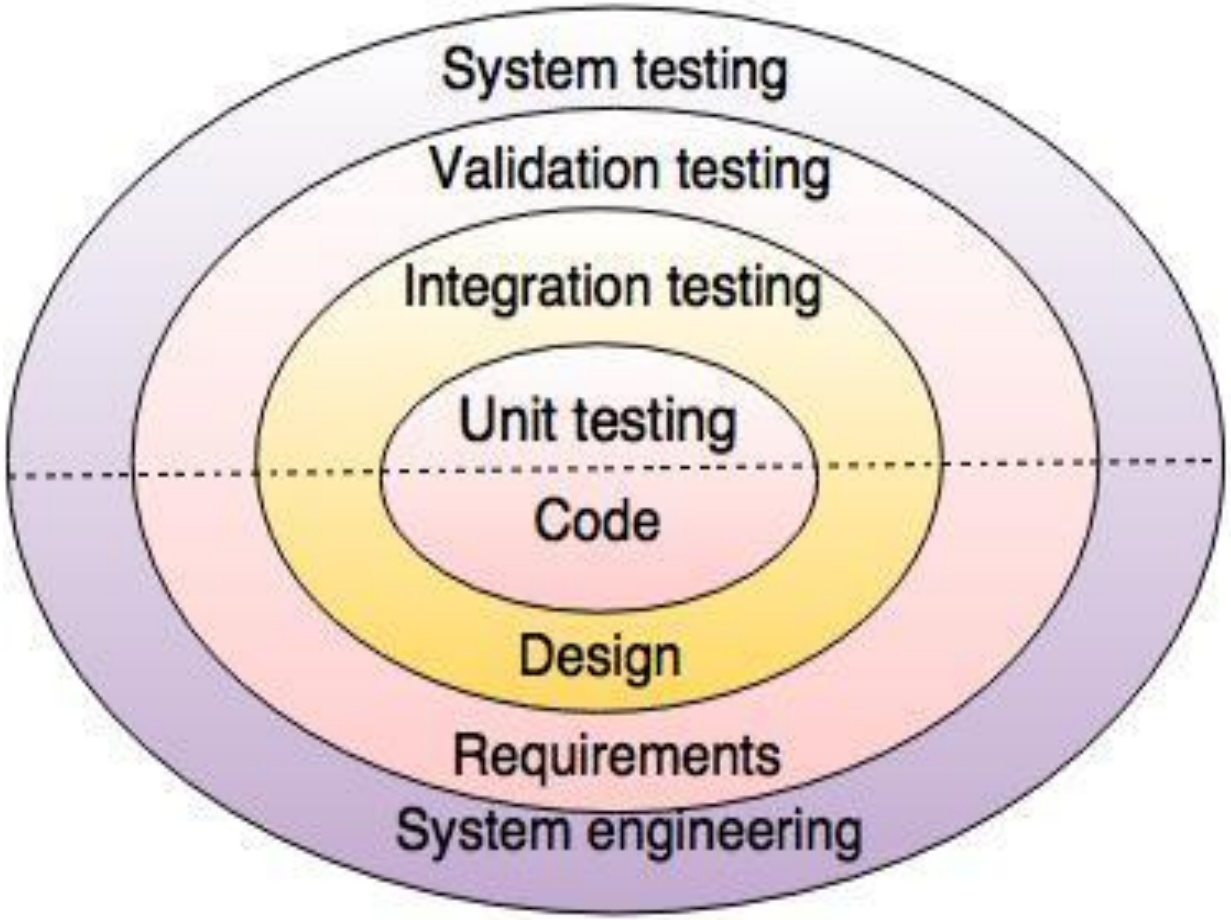

Fig. - Testing Strategy

(CONT.)

- **Unit testing:-** Unit testing starts at the centre and each unit is implemented in source code.

- **Integration testing:-** An integration testing focuses on the construction and design of the software.

- **Validation testing:-** Check all the requirements like functional, behavioural and performance requirement are validate against the construction software.

- **System testing:-** System testing confirms all system elements and performance are tested entirely.

1. UNIT TESTING

- **Focus on testing each module or software component independently based on implementation & compare the actual results with the results.....**
- **Test strategy conducted on each module interface to access the flow of input and output.**

- **Why to TEST each UNIT INDEPENDENTLY ?**
- **Because of FAULT tolerance & DEBUGGING becomes easier during unit testing.**

(CONT.)

- **What to TEST during UNIT TESTING ?**

- **MODULE INTERFACE:- Tested so that we check the correct flow of information in and outside the module.**
- **LOCAL DATA STRUCTURE:- Examine to ensure that the intermediate results or data is maintained or stored properly.**
- **Independent/ Basis/ Paths:- These are tested to ensure all statements within the modules are executed atleast once during testing.**
- **Boundary Conditions:- Output value or computation at boundary value is correct.**
- **Internal Logic:- Loops, Precedence, comparison of data types.**
- **Error Handling Paths:- Try, Catch, Statement**

(CONT.)

- **PROBLEMS with UNIT TESTING:-**

- **How to test component without anything to call it ?**
- **How to test component without any module to be called by itself ?**

- **Use DRIVER & STUBS**

(CONT.)

- **Driver:- Main program that accepts test case data, passes data to the component to be tested and prints relevant results.**
- **Take data from user & pass to test module.**
- **STUBS:- Subordinate modules that are called by the module to be tested.**
- **It is a dummy sub-program that does minimal data manipulation, provides verification of entry and returns the control to module under testing.**
- **SCAFFOLDING:- The overhead code**
- **Refers to drivers & stubs**
- **Refers to a Test Harners**

ANOTHER DIAGRAM IS DONE IN CLASS

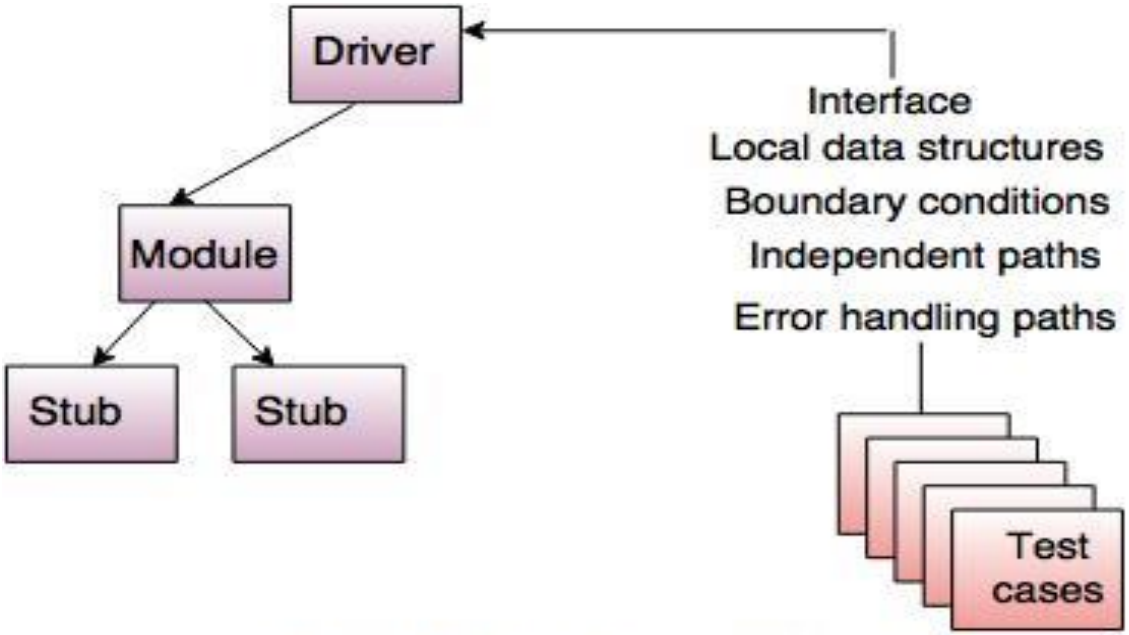

Fig. - Unit test environment

2.INTEGRATION TESTING

- **MAIN TARGET:- Determines the correctness of the interface.**
- **The units or modules are to be integrated which gives raise to integration testing.**
- **It is to verify the functional, performance, and reliability between the modules that are integrated.**

- **WHY INTEGRATION TESTING IS REQ:-**
- **Data may be lost during interfacing**
- **Global data share b/w diffn modules**
- **Sub functions may not work properly when combined.**

(CONT.)

METHOD TO INTEGRATE & TEST:-

- **Unit tested components are taken one by one and integrated incrementally -> Debugging & fault isolation becomes easier.**

Types of Integration Testing:

- **Top Down Integration**
- **Bottom Up Integration**
- **Hybrid Integration (SANDWICH)**

(CONT.)

Top-down integration

- It is an incremental approach for building the software architecture.
- It starts with the main control module or program.
- Modules are merged by moving downward through the control hierarchy.
- Calling module is always available.

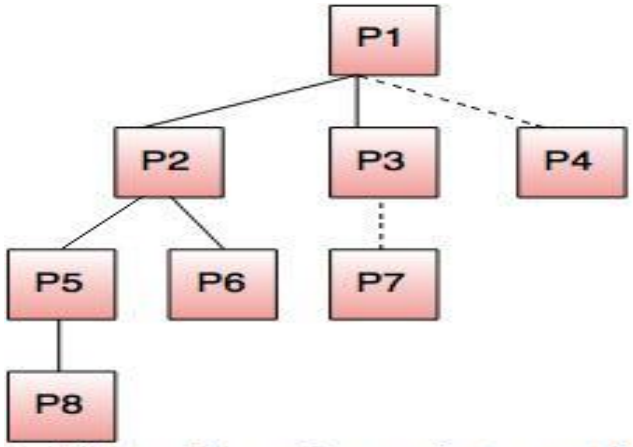

Fig. - Top-down integration

(CONT.)

Problems with top-down approach of testing:-

- It is an incremental integration testing approach in which the test conditions are difficult to create.
- Errors occur, then correction is difficult.
- The programs are expanded into various modules due to the complications.
- Previous errors are corrected, then new get created and the process continues(Infinite Loop).

BOTTOM-UP INTEGRATION

- In this the components are combined from the lowest level in the program structure.

- Components are merged into clusters then perform a specific software sub function.

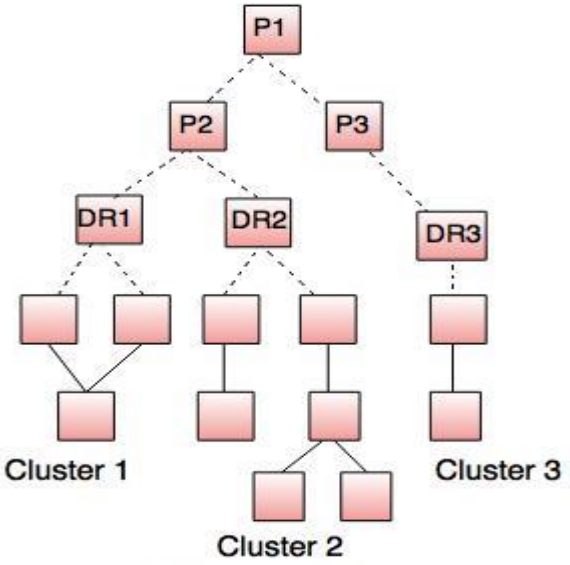

Fig. - Bottom-up integration

- A control program for testing(driver) coordinate test case input and output.

- After these steps are tested in cluster.

- The driver is removed and clusters are merged by moving upward on the program structure.

3. VALIDATION TESTING

- Done at the end of the development process to determine whether software meets the customer expectations and requirements.
- Execution of code is comes under Validation.
- It determines whether the software is fit for use and satisfy the business need.
- Includes all the dynamic testing techniques.
- It focus on user-visible actions and user recognised o/p from the system.

(CONT.)

- **What does VALIDATION testing ensures:-**

- **Functionality is achieved**
- **Correct behaviour is achieved**
- **Performance constraint meet**
- **Documents are correct**
- **A deficiency list is created in case something is missing/ incorrect.**

SYSTEM TESTING

- When the entire software system is ready.
- It verify the system against the specified requirements.
- While carrying out the tests, the tester is not concerned with the internals of the system but checks if the system behaves as per expectations.

REGRESSION TESTING & SMOKE TESTING

- In regression testing the software architecture changes every time when a new module is added as part of integration testing.

- The developed software component are translated into code and merge to complete the product.

DATA DICTIONARIES

- Data Dictionaries are simply repositories to store information about all data items defined in DFD.

Includes :

- Name of data item
- Aliases (other names for items)
- Description/Purpose
- Related data items
- Range of values
- Data flows
- Data structure definition

NOTATION MEANING

- x= a+b x consists of data element a & b

  x={a/b} x consists of either a or b

  x=(a) x consists of an optional data element a

  x= y{a} x consists of y or more occurrences

  x={a}z x consists of z or fewer occurrences of a

  x=y{a}z x consists of between y & z occurrences of a{

VERIFICATION

- ***The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.***

- **Verification will help to determine whether the software is of high quality, but it will not ensure that the system is useful. Verification is concerned with whether the system is well-engineered and error-free.**

- **It does not involve executing the code.**

- **Verification is to check whether the software conforms to specifications.**

VALIDATION

- ***The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.***
- ***Ensure that the product meet the client needs***
- **Validation is the process of evaluating the final product to check whether the software meets the customer expectations and requirements. It is a dynamic mechanism of validating and testing the actual product.**
- **It always involves executing the code.**
- **Validation is to check whether software meets the customer expectations and requirements.**

SOFTWARE METRICS

- Pressman explained as “A measure provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of the product or process”.

- Measurement is the act of determine a measure

- The metric is a quantitative measure of the degree to which a system, component, or process possesses a given attribute.

- Fenton defined measurement as “ it is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules”.

CATEGORIES OF METRICS

- **i. Product metrics: describe the characteristics of the** product such as size, complexity, design features, performance, efficiency, reliability, portability, etc.

- **ii. Process metrics: describe the effectiveness and** quality of the processes that produce the software product.

- Examples are:
  - effort required in the process
  - time to produce the product
  - effectiveness of defect removal during development
  - number of defects found during testing
  - maturity of the process

(CONT.)

- **Project metrics: describe the project characteristics** and execution.

- Examples are :
  - number of software developers
  - staffing pattern over the life cycle of the software
  - cost and schedule
  - productivity

ALPHA & BETA TESTING

- Performed by the tester who are usually internal employees of the organisation.
- Performed at the developer site.
- Reliability, security & robustness are not checked in it.
- Long execution cycle may be required.

- performed by the client & end users who are not employee of the organisation.
- Performed at the client location.
- Reliability, security & robustness are checked in it.
- Only few week of execution are required.

Alpha testing	BETA testing

BLACK & WHITE BOX TESTING

- Black box testing is the Software testing method which is used to test the software without knowing the internal structure of code or program.
- This type of testing is carried out by testers.
- Implementation Knowledge is not required to carry out Black Box Testing.
- Programming Knowledge is not required to carry out Black Box Testing.

- White box testing is the software testing method in which internal structure is being known to tester who is going to test the software.
- Generally, this type of testing is carried out by software developers.
- Implementation Knowledge is required to carry out White Box Testing.
- Programming Knowledge is required to carry out White Box Testing.

(CONT.)

- Testing is applicable on higher levels of testing like System Testing, Acceptance testing.
- Black box testing means functional test or external testing.
- In Black Box testing is primarily concentrate on the functionality of the system under test.
- The main aim of this testing to check on what functionality is performing by the system under test.

- Testing is applicable on lower level of testing like Unit Testing, Integration testing.
- White box testing means structural test or interior testing.
- In White Box testing is primarily concentrate on the testing of program code of the system under test like code structure, branches, conditions, loops etc.
- The main aim of White Box testing to check on how System is performing.