Requirement Engineering

The Process to gather the software requirements from client . analyze and document them is known as requirement Engineering

Goal → to develp and maintain sophisticated and descriptive ''system Requirement specification 'document.

Requirement Engineering Process

→ output is feasibility Study Report all the info. whether product shou be develo ornot!

- feasibility study
- Requirement ~~Engineering~~ Gathering.
- Software Requirement specification
- software Requirement validation

→ SRS is a document created by system analyst after the requirements are collected from various stakeholders.

SRS should contain following features :

- Design description should be written in pseudo code
- Conditional and mathematical notations for DFDs.
- format of forms & GUI screen prints

- user Requirements are expressed in natural language

Software Requirement validation

After requirement specifications are developed, the requirements mentioned are validated. user might ask for illegal, impractical sol^n or the requirement experts may interpret incorrectly.

-> Requirements are checked against following conditions.

- If they canbe practically implemented
- If they are complete
- If they canbe demonstrated
- If they are valid as per functionality and domain of software
- If there are any ambiguities.

Requirement ~~Software~~ Elicitation Technique

Interviews

Surveys

Questionnaries

Task Analysis

Domain Analysis

Brainstorming

Prototyping

observation

Software Requirement Specification & Characteristics

- Clear
- Correct
- consistent
- Coherent
- Comprehensible
- modifiable
- verifiable
- Prioritized
- Unambiguous
- Tracebable
- Credible Source

functional Requirements

Requirements which are related to functional aspect of software fall into this category.

Ex:- Search option given to user to search from various invoices

They define functions and functionality within and from the software system

Non-functional Requirements

- security
- Logging
- storage
- Configuration
- Performance
- cost
- gnteroperability
- flexibility
- Disaster recovery
- Accessibility

Data flow Diagram :

- A graphical tool, useful for communicating with users, managers and other personnel.
- useful for analyzing existing as well as proposed systems
- focus on the movement of data b/w external entities and processes and b/w processes and data stores.

Provides an overview of

- what data system processes.
- what data are stored.
- User and analyst.
- Analyst and ~~S~~ system designer.
- what results are produced.

DFD elements

- Source / Sink. (External Entities) □
- Data flow →
- Data Process ○
- Data Stores = (Oval or parallel lines)

Source - Entity that supplies data to system

Sink - " " receives data from system

Rules of data flow

Data can flow from

- External entity to process
- Process to external entity
- Process to store & back
- Process to process

Data can't flow from

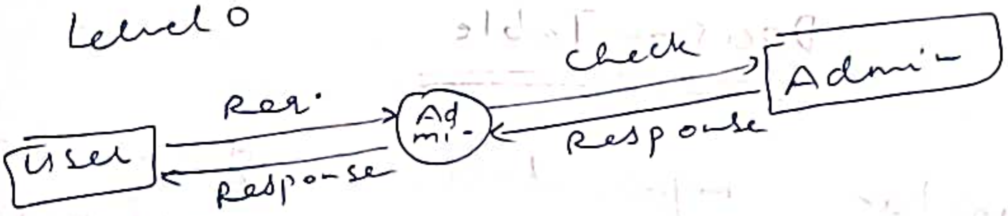
- External entity to external entity
- external entity to store
- store to store
- store to external entity.

Levels of DFD

- level 0
- level 1
- level 2

Admission to university

Level 0

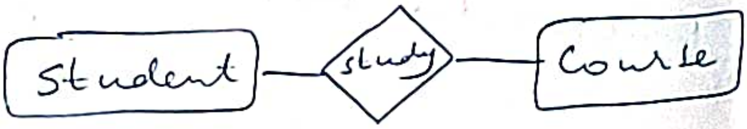


Any thing or object ← Entity

Entity Relationship Diagram

Student (Rollno. , age, address)

↓
Entity type (schema)



implemented by SQL(structure query language).

Decision Table

- * Tabular representation of conditions and their respective actions.

- * Used in ~~represent~~ requirement management as well as in system testing for checking the behaviour of different input combinations;

eg:- ATM withdrawl (circumstances)

(circumstances or) Rules T/F

Conditions	R1	R2	R3
withdrawl ≤ Balance amount	T	F	F
credit Granted	T	T	F
Actions			
withdrawl Granted	T	F	F

Test case for R1

balance = 200, requested withdrawl =

result = 'withdrawl granted'

Test case for R1

balance = 100, requested amount = 20

result = 'withdrawl denied'

Test case for R3

balance = 100, requested withdrawl = 200

no credit
result = withdrawl denied

Steps:

1. Identify requirements & define conditions
2. Define Actions
3. Define Rules
4. Define rules for action also
5. Generate Test Cases;

IEEE Standard for SRS

1. Introduction

- Purpose → Identifies product & application domain
- scope
- Definitions, acronyms, abbreviations
- Reference documents
- overview → describes contents & structure of the remainder of SRS

2 overall Description

- product perspective ← describes all external interfaces: system, user, software, also hardware, operations and site adptio and hardware constraints
- Product functions ← summary of major function
- User characteristics
- Constraints ← Anything that will limit the developers options (e.g. regulation, reliability, critically hardwar limitations, parallelism, etc)
- Assumptions & Dependencies

3 - specific Requirements

All the requirements go in here (ie, this is the body of the document). IEEESTD provides 8 different templates for this section

→ functional Requirement

→ use cases

→ External Interface Requirements

- Logical database requirement

→ Non-functional Requirements

- Appendices
- Index

SRS

- SRS is a description of software system to be developed
- It lays out functional and non-functional requirements of the software to be developed

Software Design

- The Purpose of Design ~~is~~ phase in the software Development Life cycle is to produce a solution to a problem given in the SRS (software Require-ment specification) document

- The output of the design phase is software Design Document (SDD).

- Design is a two part iterative process.

Two types of software design

1. conceptual design
2. Technical design

Basic Issues in software design

① Modularization
② Coupling
③ Cohension

Modularization :

It is the process of dividing a software system into multiple independent modules.

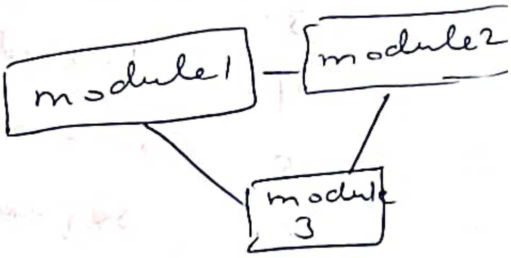
Each module works independently.

A module can be used many times as their requirements. No need to write it again and again.

Coupling

- Coupling is the measure of the degree of interdependence b/w the modules

- A good software will have low coupling.



Types of coupling

1) Content coupling.

2) Common coupling.

3) Control coupling.

4) stamp coupling.

5) Data coupling.

Cohension

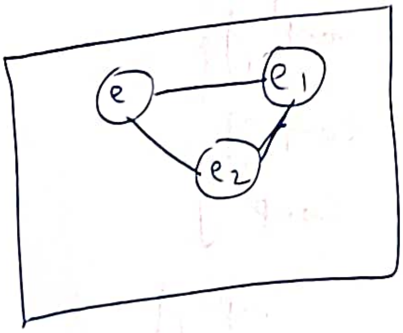
- Cohension is a measure that defines the degree of intra-dependability within elements of a module.

- It means "things that belong together should be kept together."

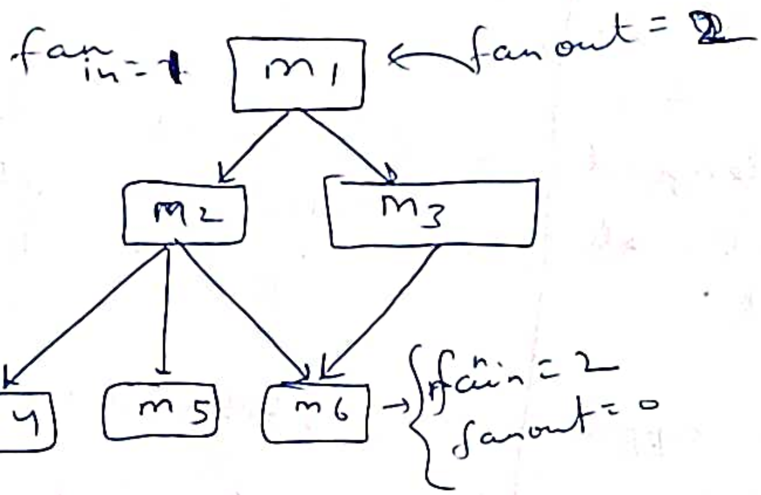
- The greater the cohension, the better is the program design.

Types of cohension

1. Location Cohension
2. logical Cohension
3. Temporal Cohension
4. Procedural Coohension
5. Communication Cohension
C. Sequential Cohension.



module structure



Cohesion

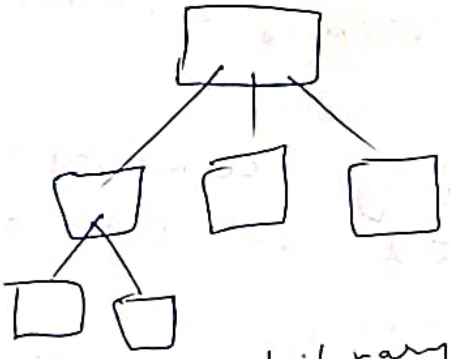
1. It is concept of intra-module.
2. represents the relationship within a module
3. Increasing cohesion is good for software.
4. Cohesion represents functional strength of modules
5. It is created b/w the same module
6. In cohesion the module focuses on a single thing
7. Highly cohesive gives best softw-are

Coupling

1. It is concept of inter module
2. represents the relationship b/w modules
3. Increasing coupling is avoided for software
4. Coupling represents the interdependence among modules.
5. It is created b/w two different module
6. In coupling the modules are connected to other modules
7. Loosely coupling gives best software

function oriented Design

- System is designed from a functional viewpoint.
- Top-down approach.
- divide and conquer approach
- DFD is used



Ex: library management system

issue design

object oriented Design

- system is viewed as a collection of objects (i.e entities)
- Bottom-up approach
- UML is used

Ex:- library management system

student faculty B

Unified modelling language

S/w quality Assurance

* It is a process which ensures that developed s/w meets and complies with defined or standarized quality specifications .

  Ex: ISO-9000, SEI-cmm etc.

* It incorporates all s/w development process starting from defining requirements to coding until release

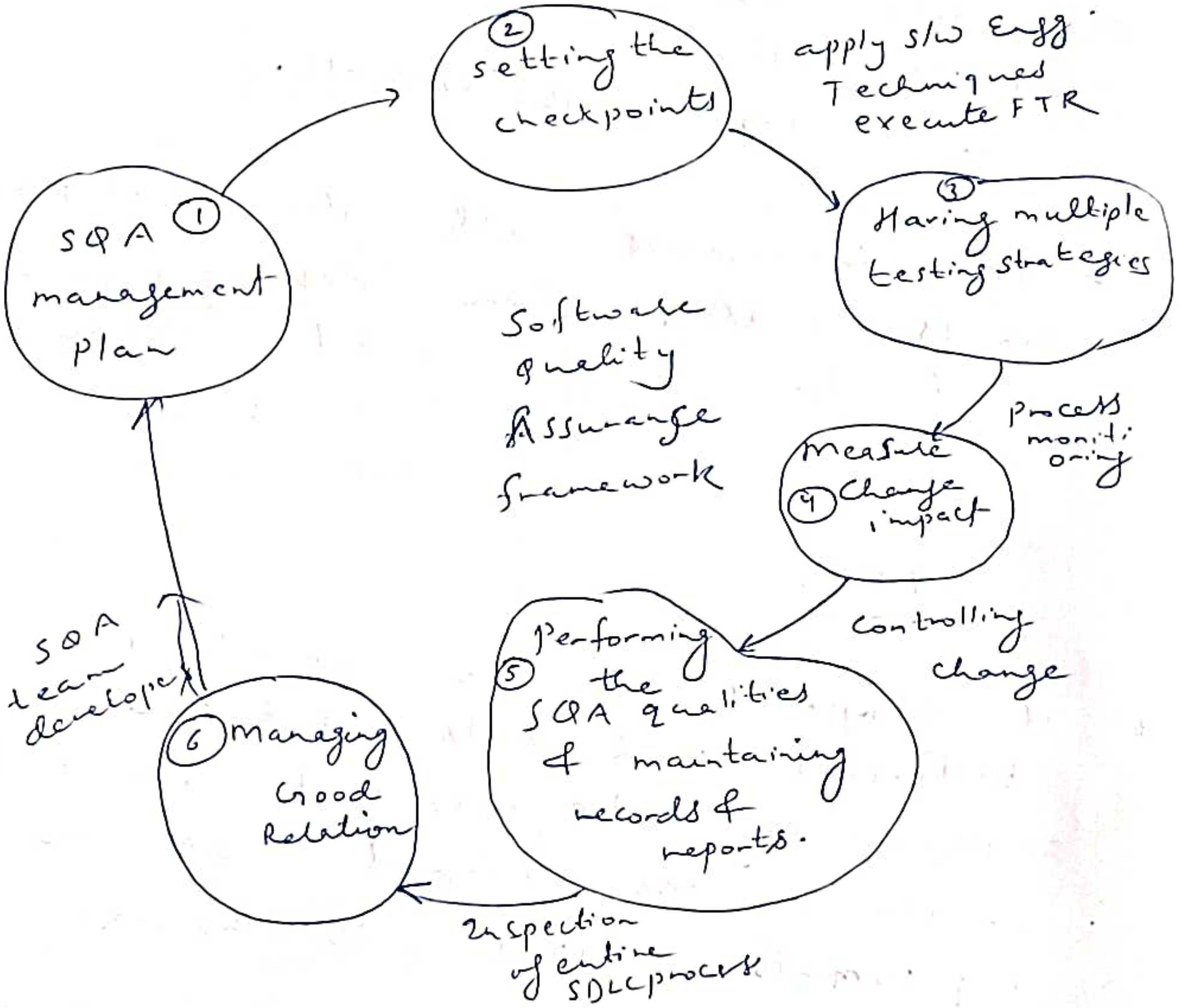
SQA Plan

* It comprises of the procedures, techniques and tools that are employed to make sure that a product or services are aligned with the requirements defined in the SRS

* SQA Plan identifies → SQA responsibilities of a team
  → List of areas need to be reviewed & audited

P

SQA frameworks /Activities



② setting the checkpoints
apply s/w Engg. Techniques execute FTR
③ Having multiple testing strategies
process moniti oring
Measure ④ Change impact
controlling change
⑤ Performing the SQA qualities & maintaining records & reports.
Inspection of entire SDLC process
⑥ Managing Good Relation
SQA team developer
SQA ① management plan
Software quality Assurance framework

FTR — formal Technical Review

TOP DOWD APPROACH

(1) In this approach we focus on breaking up the problem into smaller parts

(2) Mainly used by structured programming language such as COBOL, Fortran, c etc

(3) each part is programmed separately therefore contain redundancy.

(4) Communication is less amping modules

(5) It is used in debbuging, module documentation etc.

(6) In this decomposition takes place

BOTTOM UP APPROACH

(1) In bottom up approach we solve smaller problems and integrate it as whole and complete the solution

(3) Mainly used by object oriented programming language such as C++, C#, python.

(3) Redundancy is minimized by using data encapsulation and data hiding

(4) In this module must have communication.

(5) It is basically used in testing

(6) In this composition takes place.

DFD	ERD
① It stands for Data flow Diagram	It stands for Entity Relationship Diagram or model
2. Main objective is to represent the process and data flow b/w them.	main objective is to represent the data object or entity and relationship b/w them.
③ It explains the flow and process of data input, data output and storing data	③ It explains and represent the relationship b/w entities stored in a database.
Symbols used in DFD are:	symobls used in ERD are:
rectangles (represent the data entity)	rectangles (represent the entity)
circles (~~process~~ represent the process)	diamond boxes (represent relationship)
arrows (represent the flow of data)	lines and standard notation (represent cardinality)
ovals or parallel lines (represent data storing)	

(4) It models the flow of data through a system	(4) It model entities like people , objects, places and events for which data is stored in a system.

functional Requirements

① A functional requirements defines a system or its component

② It specifies "what should the software do?"

③ functional requirement is specified by user.

④ It is mandatory.

⑤ It is captured in use case.

⑥ Defined at a component level
usually easy to define.

⑦ Ex:- System shutdown in case of cyber attack

⑧ Helps you verify functionality of the software

Authentication of user whenever he/she logs into the system

Non-functional

① A non-functional requirement defines the quality attribute of a software system

② It places constraints on "How should the system software fulfill the functional requirements?"

③ Non-functional requirement is specified by technical peoples
eg: Architects, Technical leaders and software developers

④ It is not mandatory

⑤ It is captured in quality attribute

⑥ Helps you verify the performance of the software functional Testing like System, integration, End to End, API testing etc. are done.

Non-functional testing like performance stress, testing etc. are done ✓

Ex:- The site should load in 3 sec when the no. of simultaneous users are > 1000 ✓

Authentication of user whenever he/she logs into the system ✓

Application

✓ Product definition is stable
✓ There is no ambigous requirement
✓ The Project is short

Software requirement Specification (SRS) format :-

~~SR'S as the name suggests, is complete specification and description of requirements of software that needs to be fulfilled for successful development of software system. These requirement can be functional or non-functional depending upon type of requirement.~~ ✓

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of ~~requirements~~ engineering phase ✓

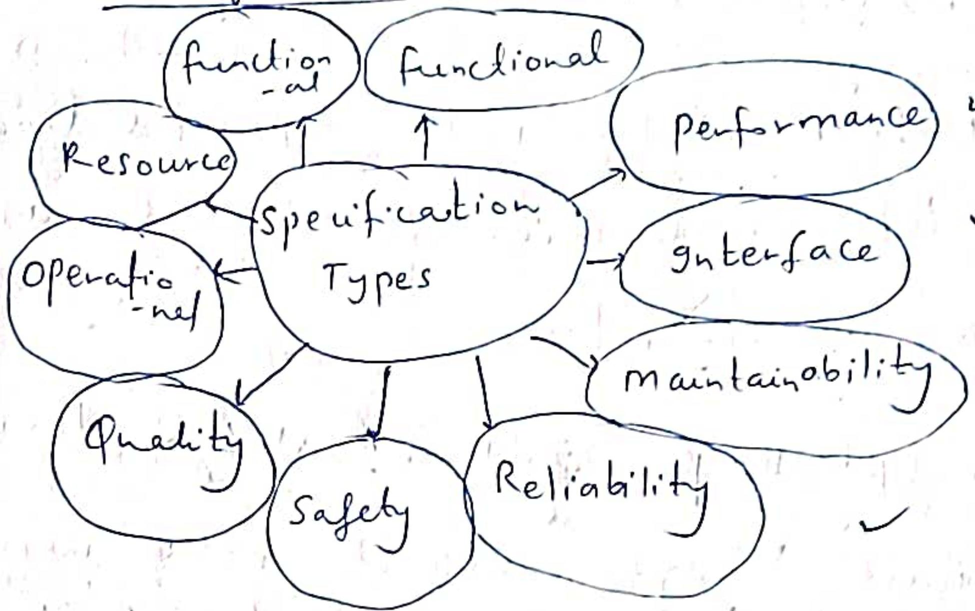
Modifiable
Traceable
variable

Qualities of SRS :-

✓ Correct ✓
✓ Unambigous ✓
✓ Complete ✓
✓ consistent
✓ Ranked for importance / stability
✓ varifiable
✓ Modifiable
✓ Traceable

Unambigous

Types of Requirements



Software Design

The design phase of software development deals with transforming the customer requirements as described in the SRS documents. ~~in the SRS~~ ~~documents~~ into a form implementable. using a programming language.

1) Interface Design

2) Architectural Design

3) Detailed Design.

Interface Design

Interface design is the specification of the interaction b/w system and its environment.

this phase proceeds at a high level of abstraction with respect to the inner workings of the system ie, during interface design, the internal of the systems are completely ignored and the system is treated as a black box.

It should include following details:-

- Precise description of events in the environment, or message from agents to which the system must respond.

- Precise description of the events or message that the system must produc

- specification on the data and the formats of the data coming into and going out of the system.

- specification of the ordering and timing relationships b/w incoming events or messages, and outgoing events or outputs. ✓

Architectural Design

Specification of the major components of a system, their responsibilities, properties, interfaces and relationships and interaction b/w them. ✓

- Allocation of functional responsibilites to components
- Component interfaces
- Communication and iteraction b/w components.
- Component scaling and performance propertres resource consumption properties, reliability properties, and so forth ✓

Detailed design

Design is the specification of the internal elements of all major system components, their properties, relationships, processing and often their algorithms and the data structures.

The detailed design may include

- Algorithms and data structure
- User Interfaces
- Unit states and states changes
- Data and control interaction b/w units.

ISO 9000

ISO stands for Internationas organisation for standarisation. It is an indepedent non-yonernment internationas organsation for developing software.

These standards are developed for ensuring quality. Safety. efficiency of product , services and systems.

ISO 9001, ISO 9002 ,ISO 9003

It is a series of standards developed by ISO .. These standards haste been developed for ensuring the quality for manufacturing and Service industry.

Initially, this series have been launched ISO 9001, 9002 and 9003

After Sometime it also launch the ISO 9004

ISO 9000:

It comes in 1987
It is defined as set of international standard on quality manegement and quality ensurance it if applicable to any size of organisation aid this standard usable by all sectors -

ISO 9001

This standard is provide to the organisation which are involved in creating new products. It focuses on quality esurance in design, development and production and software development organasation also.

ISO 9002

This standard is applicable to those companies which donot design product but involved in manufacturing Ex:- car and steel manufacturing company.

ISO 9003

This standard is applicable to the organisation that are only involved in installation and testing of the product

ISO 9004

This is the latest version of ISO 9000 Series this standards gives

tion guidelines for

for enchancing an organisation ability to achieve a sustain success to develop a quality product.

It provide a self assessment tool