Module - 3

→ What is Requirement?

- A software req. is a property which must be exhibited by s/w developed or adopted to solve a particular ~~property~~ problem.
- The req. for a system are the description of the services provided by the system & it operational constraints.
- It depicts needs of customers.

Requirements Engineering

The process of finding out, analysing, documenting and ~~de~~ checking these services and constraints is called Req. Engineering.

Types of Requirements

1) User Reqt.
2) System Reqt.
3) Software Specification

Categories of Reqt.

1) Functional Reqt.

It describes what the system should do These reqts. depend on type software being developed. The expected user of the s/w & the general approach taken by the organization when writing reqt.

In principle, the functional reqt. specification of system should be both complete & consistent

DELTA Pg No.
Date / /

2.) Non-functional requirements.
The reqt. that are not directly concerned with the specific functions delievered by system.

Classification of Non-functional reqt.

a) Product Requirements - such as reliability, efficiency, executional speed, etc.

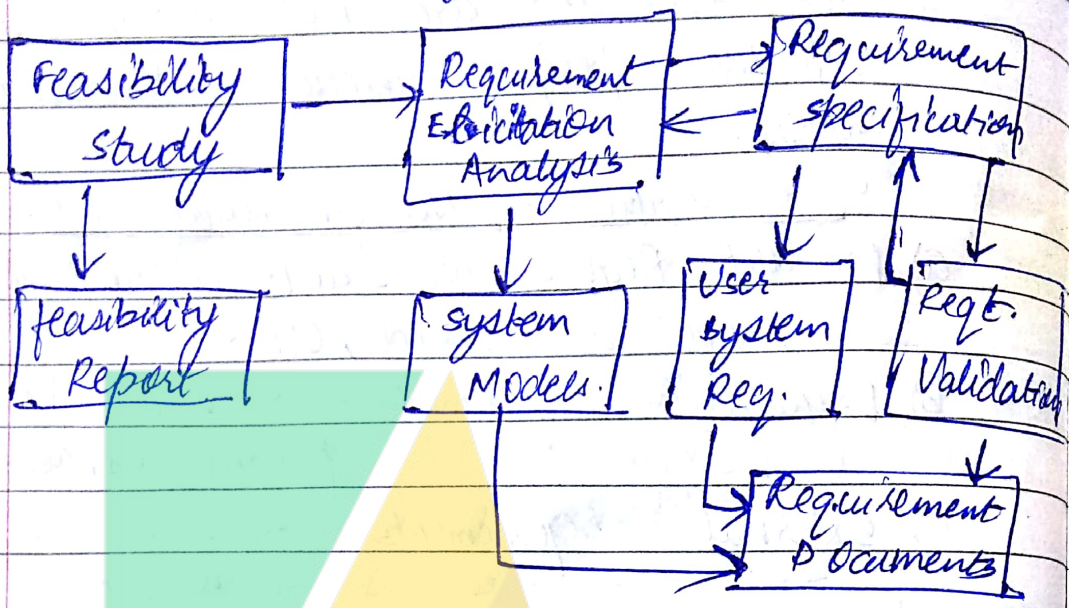
b) Organisation requirements - such as policies & procedures, delievery, standard reqts, etc.

c) External requirements - such as inter-operability safety, privacy, etc

3.) Domain Reqt.
Domain requirements are desived from the application domain of the system. Rather than from the specific needs of the system users

DELTA Pg No.
Date 10 / 09 / 2018

Requirement Engineering Process



(1) Feasibility Study

It decides whether or not the proposed system as worth while. The inputs to the feasibility study is a set of preliminary business req., an outline description of system and how the system is intended to support business process

It involves:-

a) Information assessment
b) Info. collection.
c) Report writing

Different aspect of conduction feasibility study

i) Economic Feasibility - It gives the top management the economic justification for the new system

ii) Technical feasibility - Understanding the different technologies involved in proposed system.

iii) Legal feasibility - The development of certain sensible system may have some

Leagal & ramification.

iv) Organisational feasibiletey - In this developer gets the information about man power resources and work culture of the organisation.

Documentation / Format of Feasibiliby study

1. Introduction
   1.1. Statement of problem
   1.2 ~~Impact~~ Implementation ~~of Requirement~~ Environment
   1.3 Constraints.

2. Management summary & Recomendation
   2.1 Important findings
   2.2 Coments
   2.3 Recommendations
   2.4 Impact

3. Alternatives
   3.1 Alternative system configurations.
   3.2 Criteria used in selecting the final approa

4. System Description.
   4.1. Abbreviated state of scope
   4.2 Feasibility of arrogated elements

5. Cost Benefit Analysis.
6. Evaluation of Technical Risk
7. Leagal Ramification
8. Other project specific problem.

TOT Classes

Numerical

Q-1

Functional Units	Weighing factors			
	Low	Avg	High	
EI	3	4	6	40
EO	4	5	7	30
EQ	3	4	6	20
ILF	7	10	15	5
EIF	5	7	10	7

Functional units with weighing factors = average

$F.P = UFP \times CAF$

$CAF = 0.65 + (0.01 \times \sum f_i) \rightarrow 42$

$= 0.65 + 0.42 = 1.07$

$UFP = 40\times4 + 30\times5 + 20\times4 + 5\times10 + 7\times1$

$= 489$

$\sum f_i = 14 \text{ questions} \times 3 = 42$

$F.P = 489 \times 1.07$

$= 523.23$

Putnam Resource Allocation Model

Introduction:- This model assumes that effort during program development follows a Rayleigh type curve.

The number of project members per unit of time modelled by the Rayleigh curve equation.

$$\frac{dy(t)}{dt} = 2Kat e^{-at^2}$$

where $\frac{dy}{dt}$ is the personal utilisation rate in persons per month

K is total lifecycle effort from the start of the project through maintenance

a - is a parameter that affects the shape of the curve.

t - is elapsed time.

Q-2 Using cocomo model the KLOC is 60. Identify mode of project. Calculate

→ t due, effort, avg staff size & productivity

Sol semi detached

$$E = a_b (KLOC)^{b_b}$$
$$= 3(60)^{1.12} = 294.2$$

$$t_{due} = C_b (E)^{d_b}$$
$$= 2.5(294.2)^{0.35}$$
$$= 18.28$$

$$\text{Avg staff size} = \frac{E}{T_{due}} = 16.09$$

Productivity = 0.2

DELTA Pg No.
Date 12/09/2018

(2) Requirement Elicitation Techniques

Meaning - 1) It is the activity during which software requirements are discovered, articulated & revealed from stake holders

2.) It includes the fol. activities
a) Knowledge of general area where the system is applied
b) The details of the specific customer problem where the system will be applied must be understood
c) interaction of system with external requirements
d) Detailed investigation of Users Needs.
e) Define the constraints of system development

Process:
1. Identify all stakeholders
2. Ask relevant questions in order to gain an understanding of the problem.
3. Analyse the information.
4. Confirm your understanding of Reqt. with the stakeholders
5. Create requirement statements

Techniques:-
1) Interviews
2) Brain-storming
3) Facilitated Application Specification Technique (FAST)
4) Task Analysis
5) Quality function deployment
6) Use Case Based Req. elicitation

(3) Requirement Analysis

1. After performing Reqt. gathering there comes a phase - Requirement analysis.
2. During Reqt. analysis, the following task are preferred.

a) Reqt are ~~that~~ categorised & organised into respective ~~to~~ subsets

b) The relationship among reqt. is established.

c) The requirements are further examined for checking their consistency.

d) Finally reqt. are ranked as per user needs

i) Draw the context diagram - It is a simple model that defines boundaries and interfaces of the proposed system with the external world.

ii) Development of a Prototype - It helps the client to visualise the proposed system and increase the understanding of reqt.

iii) Model the requirements - It consists of various graphical representation of it Such models includes:- DFD's, ERD's, Data Dictionaries, etc.

iv) Finalize the reqts. - After modeling the requirements we will have better understanding of system behaviours.

Date / /

(4.) Requirements Documentation

Documentation is also called requirement specification. It is the activity during which requirements are recorded in one or more forms usually in a SRS

(5.) Requirement Validation

The objective of R.V is to certify that the SRS document is an acceptable document of the system to be implemented.

17/09/18

Software Requirement specifications (SRS)

SRS is a document that is generated as an o/p of requirement analysis. It is an agreement b/w the developes and the customers.

Problem without SRS | Imp. of SRS :-

Without SRS, the system would not be implemented ~~as~~ according to customer needs.

~~Software Requirement specification~~

Without SRS, it is very difficult for the maintainence engineers to understand the functionality of the system

Goals of SRS:-

1. Feed back to customer
2. Problem decomposition.
3. Input to design specification
4. Production validation check.

Who Writes SRS documents?

A requirement gathering team consisting solely of programmers, product marketers, system architects and the project manager aims the risk of creating a SRS doc.

Properties of SRS:-

1. Concise
2. Structured
3. Black Box View
4. Response to Undesired events
5. Integrity
6. Verifiable.

Characteristics of SRS document:-

i) ~~Cott~~ correctness - A SRS is correct if every requirement included in the SRS represent something required in the final system.

ii) Completness - The SRS is complet when it is documentd after
   a) correct defination of scope
   b) focusing on all problem/goals/objective.

3. Unambiguous - If an only if every requirement stated has an end only interpretation.

4. Verifiable - An SRS is verifiable if and only if there exist some cost effective process that can check whether the final product meets the requirement.

5 Modifiable - An SRS is modifiable if its structure & style are such that any neccessary change can be made easily

6. Traceable - An SRS is traceable if the origin of each of the requirement is clear and if it facilitates the referencing of each requirement in future development

7. Consistency - Consistency in SRS is essential to achieve correct results across the system.

8. Testibility - SRS should be written in such a way that it is possible to create a test plan to confirm whether specifications can be meant.

9. Clarity - An SRS is clear when it has single interpretation for the author, user, stake-holder, developer, tester and the customer.

10. Feasibility - It needs to be confirm whether the technology is capable enough to deliever what is expected in SRS

⇒ Parts of SRS

1. Functional requirements
2. Non-functional requirements
3. Goals of SRS

Information flow Matrix

It is also called IF Matrix. IF Matrix model the degree of Cohension and coupling for a particular system component.

a) Any element identified by decomposing a system into diffesent parts.
&
b) Cohesion :- The degree to which our component performs the single function
c) Coupling - It is used to describe the degree of linkage between 1 component through other in the same system.

For eg:- For a component A, we can define
a) Fan in : Accounts the no. of other components that can call or passed to component A.
b) Fan Out : Is the no. of components that are called by Component A.

Steps of IF Matrix :-

1) Level of each component in the system design.
2) For each component count the no. of calls to that component. This is the Fan-In of that component
3) For each component, Count the no. of calls for that component.
4) Calculate the IF value for each component

DELTA Pg No.
Date 18/09/2018

Components of SRS

There are 9 components that must be addressed when designing & writing SRS

1. Interfaces
2. Functional Capabilities
3. Performance Levels
4. Data structure Elements
5. Safety
6. Reliability
7. Security
8. Quality
9. Constraints

Uses of SRS

1. Project managers base their plans and estimates of schedule effort and resources on it.
2. Development team needs to develop product
3. Customer rely on it know what product they can expect
4. Testing group needs it to generate test plan.

Benefits of SRS

1. Establish the basis for agreement b/w the customer & supplier on what the s/w product is to do.
2. Reduce the development effort
3. Provide a basis for estimating cost & schedule.
4. Provide a base line for validation & verification

5. It serves as a basis of later enhancements of finished product.

Format / Template of SRS

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Definition & acronyms
   - 1.4 Reference
   - 1.5 Overview

2. Overall description
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User characteristics
   - 2.4 Compulsion
   - 2.5 Assumptions & dependencies

3. Special Requirements.

Appendices

Index

2.1 Product perspective

2.1.1 System interfaces

2.1.2 Interfaces

2.1.3 Hardware interfaces

2.1.4 software interfaces

2.1.5 communication interfaces

2.1.6 Memory Constraints

2.1.7 Operation
2.1.8 Site Adaptation Reqt.

3.1 External Interfaces.
3.2 Functions
3.3 Performance Requirements
3.4 Logical Database Reqt.
3.5 Design constraints
3.6 Software System Attributes
3.7 Organising the Specific Reqt.
3.8. Additional Comments

Benefits of ISO 9000

1. Continous Improvement
2. Improve Customer Satisfaction
3. Boost employ moral
4. Increase employ participation
5. Better product / services.
6. Eliminate variation
7. Greater Quality Insurance
8. Improve profit levels.
9. Reduce Cost
10. Improve Communication

Design :-

The design activity begin when the requirement document for the s/w to be developed is available.

The goal of the design is to represent a system which can be used later to build that system. It is the only way by which we can accurately translate the customer requirements into a finished s/w product or system.

s/w design is a process of inventing and selecting programs that meet the objectives for a software

Characteristics / Objectives :-

1.) Correctness
2) Understandibility
3) Efficiency
4) Maintainability

S.No	Characteristics	Good design	Bad design
1.	Change	Change in 1 part of system. Doesn't always require change in another part of the system	1 change requires changes to many parts of system
2.	Logic	Every piece of logic has 1 and only 1 home.	Logic has to be duplicated.

3.	Cost	Less in good design	More in Bad design.
4.	Extension	System can be extended with change in only one place	System cannot be extended easily
5	Nature	Simple ~~Complex~~	Complex

Analysis Model & Design Model.

Analysis Model

The elements of analysis model are:~~data~~

1. Data dictionaries 2. ERD (Entity Relationship diagram)
3. DFD (Data flow Diagram) 4. State-transition Diags
5. control specification 6. Process specificatio

Design Model

The elements of design Model are:-

1. Data Design
2. Architectural design
3. Interface design
4. Component level design

(1.) Data Design

It transforms the information domain model created during analysis into data structures that will be required to implement the software. The data objects and relationships defined in Entity relationship Diagram (ERD) and detailed Data Content depicted in the

data dictionary provide the basis for the data design activity

2.) Architectural Design
It defines the relationship b/w major structural elements of the software, the design patterns that can be used to achieve the requirements that have been defined for the system.

3.) Interface Design
It describes how the software communicates with itself, with systems that interoperate with it and with humans who use it.

4.) Component level design.
It transforms the structural elements of the s/w architecture into a procedural description of s/w components.

Software design Models

It is divided into 2 classes :-

1. Static Model :
It represents the various aspects of s/w that don't change during execution.
for eg:- Use case Diagram, Class Diagram.

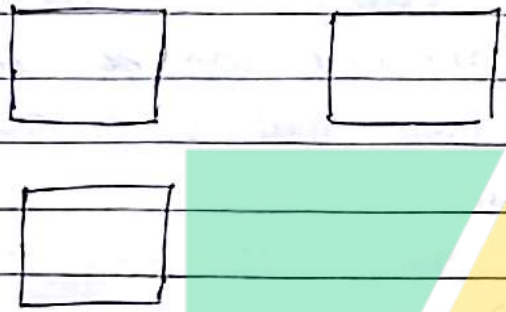
2. Dynamic Model :
It shows what happens during s/w execution. for eg:- DFD, State-transition diagram, etc.

DELTA
Date 03/10/2018

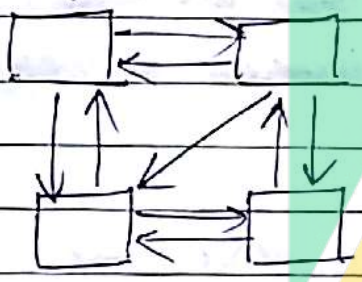
Coupling

Types :-

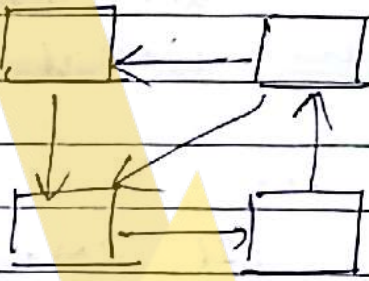
1.) Uncoupled



2.) Highly coupled



3.) Loosely coupled



Types coupling.

(Best)

Data coupling.	data is passed
Stamp coupling	DS is passed.
Control coupling	flag ('coupling is other module using goto.
External coupling	H/W (Hardware)
common coupling	sharing
content coupling	calling of inner func

(WORST)

DELTA Pg No. Date / /

Modularity

Coupling - is the measure of the degree of interdependance b/w modules.

Types of coupling:- (Types of interconnection)

1. Highly coupled - When modules are highly dependend on each other.
2. Loosely coupled - When modules are dependent on each other but interconnection among them is week
3. Uncoupled - When the different modules have no interconnection among them

Types of coupling

1. Data coupling - The dependency b/w 2 modules is said to be data coupled if their dependancies is based on the fact they communicate by only passing of data.

2. Stamp coupling - Two modules are stamp coupled if the communicate using the composite data item such as structure, object, etc.

es module
ing gold.

3. Control coupling - Module A & module B are said to be control coupled if they communicate by passing of control information.

4. External coupling - In it, a Module has a dependancy to other module external to the software being developed. or to a

particular type of hardware

5. Common Coupling — With common coupling, module A & module B have shared data making a change to the common data means tracking back to all the modules which access that data to evaluate the effect of change.

6. Content Coupling - Content coupling occurs when module A changes data of module B or when control is passed from 1 module to the middle of another module.

DELTA Pg No
Date / /

Cohesion

Cohesion is the major of functional dependencies among the elements within a single module.

A cohesion module perform a single task within a software procedure, requiring a little interaction with procedures being performed in other parts of the programs.

A strongly cohesive module implements functionality that is related to one feature of the solution and requires little or no interaction with other modules.

Types of Cohesion :-

- Functional Cohesion
- Sequential Cohesion
- Communicational Cohesion
- Procedural Cohesion
- Temporal Cohesion
- Logical Cohesion
- Coincidental Cohesion

Best ↓ Worst

DELTA

Types of Cohesion

1. Functional Cohesion - It is said to exist if all activities in the module are functionally related or they are performing a similar function, such as Interest Calculation, managing a employ Pay roll.
2. Sequential Cohesion
   In sequential cohesion, modules are divided into series of activities such that the o/p of 1 module becomes the input to the next module and the chain continues

3. Communication Cohesion.
   A module is set to have comm^n. cohesion if all the functions of the module referred or update the same data structure.

4. Procedural Cohesion
   In this type of cohesion, series of steps is to be implemented. For eg - The algorithm for decoding a message.

5. Temporal Cohesion:
   When a module contains function that are related by the fact that all the functions must be executed in ~~the~~ a some time-span

6. Logical Cohesion
   In logical cohesion activities belonging to the same categorie are grouped together,

DELTA Pg No
Date / /

we can group ~~the~~ all reporting activities together or all querying activities together.

7. Coincidental Cohesion.
In it instruction have no relationship to each other, they just coincidental feel in the same module.

Difference b/w Functional Oriented & Object Orient

Functional Oriented	Object Oriented
1) The basic abstractions which are given to the user are real world functions such as sort, merge, etc.	1.) Basic abstraction is not the service available to the user of the system but are the data abstractions where the real world entity are represented such as student, customer, etc.
2) The state information is available in a centralised shared data store.	2) The state info" exist in the form of data distributed ~~amo~~ among several objects of the system.
3.) Functions are grouped together by which a higher level function is obtained.	3. In this design the functions are grouped together on the basis of data they operate on, such data members & member fn.
4) It operates on function	4.) It operates on objects or classes
5) Functions, gets some data, process it and then return results.	5.) Objects is data with methods of processing it.

DELTA
Date: 09/10/2018

Software Design Strategies

(1.) Top down
(2.) Bottom-up

Top down

1. It is an informal design strategy for breaking problems into smaller problems.

2. The design activity must begin with analysis of requ-, definition and should not consider implementation details at first.

Importance

The essential idea of top-down design is that the specification is viewed as describing a black box in programs. The designer should decide how the internals of black box is constructed from the smaller black boxes and that those inner black boxes be specified.

In top-down design, system components are derived from project specification.

Conclusion :

In top down strategy, a S/w project is decomposed into sub projects and this procedure is repeated until the sub task have become so simple than an algorithm can be formulated as a solution.

DELTA Pg No
Date / /

This Approach is used when the specifications are clear and the development is from the scratch.

Bottom Up

1. In Bottom-up approach we identify modules that are required by many programs.
2. These modules are collected together in the form of library
3. Now we decide how to combine these modules to provide larger ones.
4. In bottom up design the approach is to start at the bottom with problems that we already know how to solve.

Note - It is used when objectives are not clear.

Modularity

Introduction - A system is considered modules if it consists of discrete components so that each component can be implemented seperately and a change to one component has minimal impact of other components

Desirable properties of a Modular System

1. Each function in each abstraction has a single well defined purpose.
2. Each function manipulates no more than 1 measured data structure.

DELTA Pg No. Date / /

Advantages :-

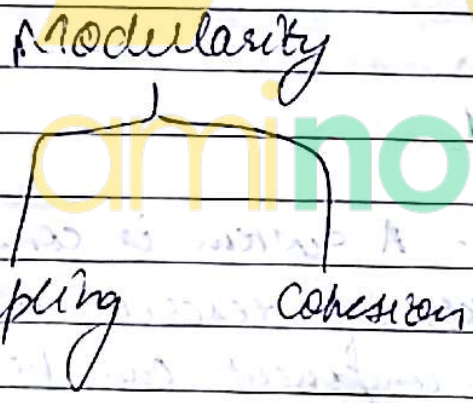
1. Modular systems are easier to understand and explain because their parts make independent functionality.

2. Modular systems are easier to document because each part can be documented as an independent unit.

3. Testing and Debugging individual modules is easier

4. Bugs are easier to isolate and understand

5 Well composed modules are more usable



DELTA Pg No.
Date 10/09/2018

Function Oriented Design

Intro - It is an approach to s/w design where the design is decomposed to set of interacting units where each unit has a clearly defined function.

In function oriented design, the result - focusing attention to the function of the program. The following diagrams are designed.

1. Data flow Diagram
2. Data Dictionaries
3. Structure chart
4. Pseudo code

Object Oriented Design

Intro - It implies that the s/w consist of a collection of objects that include data and methods

The characteristics of OOD are:-

1. Encapsulation
2. Data Abstraction
3. Inheritance
4. Polymorphism

OOD is a result of focusing attention not on the function performed by the program but instead on data that are to manipulated by the program.

DELTA PG NO. DATE / /

Steps to analyse and design OOD

1. **Create use-case model**
   To model a system, the most important step is to capture the dynamic behaviour. The first step is to identify the actors interacting with the system and then write the use-case that depicts the functionalities of the users.

2. **Draw activity diagram**
   Activity diagram shows the dynamic nature of system by modelling flow of control from activity to activity.

3. **Draw interaction diagram**
   It models the behaviour of use-cases by describing the way, group of objects interact to complete the task.
   2 kinds of interaction are:-
   i) Sequence Diagram
   ii) Collaboration Diagram.

4. **Draw the Class Diagram.**
   Class diagram is a static diagram. It represents the static view of an application. It describes the types of objects in a system and the various kinds of relationship that exist among them.

DELTA Pg No. Date / /

5. Draw State Chart Diagram :-

It is used to show the state of a class or the event that cause a transition from one state to another and the actions that result from a state change.

6. Draw component and Deployment Diagram

Component diagram address the static implementation view of the system.

They are related to class diagrams in that the component maps to one or more classes

~~They~~ The deployment diagram captures the relationship b/w physical components & hardware.

7 Write a short note on info hiding and structured programming.

(Notes from pic)