UNIT - 2

Requirements Analysis

PAGE NO :
DATE : / /

* Statement of system scope :-
Project scope is the part of project planning that involves determining and documenting a list of specific project goals, deliverables, features, functions, task, dead lines and ultimately cost.
In other words, it is what needs to be achieved and the work that must be done to deliver a project.
It is important to pin down the scope early in a project life-cycle, as it can greatly impact the schedule or cost or both of the project down the track.

* Review :- A review is a systematic examination of a document by one or more people with the main aim of finding and removing errors early in the software development life-cycle. Reviews are used to verify for document such as requirements system, design code, test plan and test cases.
Reviews are usually perform manually while static analysis of the tools

is performed using tools.
Importance of review process productivity of the team is improved and time sketch reduce repulse the correction in early stage and work products will help to ensure that those work products are clear and ambitious.
Reduction in cost because favour defects in final software. Testing costs and time is reduced as there is enough time ispend during the initial phase.

* Software Refinement:- Modern software development is a complicated process specially when a software system become large and complicated. Software developers must apply software refinement in order to procede from a high level abstract model to a final executable software system by adding more details over time. Class diagrams are important because they represent the static structure of a software system and therefore we design a UML profile extending the UML meta model to support class diagram refinement.
Refinement is an iterative process

working with a partner who is not on the design team to proof, read each other work.

* Analysing the problem — Analysing is the systematic process of reasoning about a problem and its constituent part to understand what is needed or what must be done. Analysis also involve communication with many people initially those who are most familier with the existing need and its surrounding that is the problem domain must be contacted Requirement engineers must also communicate with whose who are working completeness checking requires examination of the problem as a prereceived by those who are most familier with it so that the result will satisfy the real needs.

Key points of analysing the problem-

1- Problem analysis is the process of understanding real world problem and users needs and purposing solution and meets those needs.

2- The goal of problem analysing is to gain a better understanding before development being of the problem being solved.

3- To identify the root case of the problem behind the problem ask the people directly invoked.

4- Identify the actors on the system if a key step in problem analysis.

* Software Requirement Specification (SRS) :- A document used to describe the behaviour of software system, functional, non-functional requirements of the software system.

- Functional requirement – This is the list of the actual services which a system will provide. This is the list of the service / function which a user wants from the software.
For example – Business rules of the particular organisations for which developing software.

- Non-functional requirement – How a system should behave while performing the

DATE : / /

operations. These are the constants on the services which system is offering. For example - response time coverability.

* Users of the SRS -

1- Development team
2- Maintainance team
3- Client
4- Technical writer

* Contents of the SRS -

1- Category - What kind of your software.

2- Purpose - Describe what is the purpose of making this system (as a nothing is without any reason.

3- Scope - What is the area it covering, what is its range, in what limit it will help you.

4- Introduction - Define the existing system and purpose system.

5- Advantage - Define the advantage of the system.

* Review for correctness :- Correctness from software engineering prespective can be define as the adherence to the specification that determine how users can interact with the software and how the software should behave when it is used correctly. If the software behave incorrectly, it might take considerable amount.

1- Undetected errors remove.
2- Programming code write with programming rules.
3- Missing items correct.
4- Easy to involve other person.
5- Easy to implement.
6- Focus on system making error free.
7- Improve the quality of software.
8- Produce the result according to expectation.

* Software designing :- Software designing is the process of envisioning and design define software solution to one or more sets of problem. One of the main component of software design in the software requirement analysis (SRA) is a part of the software development process that list specification use in software

engineering. The design focus on capabilities and multiple design for the same problem can will exist.

The design process is a sequence of stack that enables the designers to describe all aspects of software for building. Creative skills, past experience, a sense of what makes good software and an overall commitment to quality are examples of critical success for a component design.

Requirement Completeness

A quality demanded to the set of software requirements and to each requirement itself, in order to ensure that there is no information left aside.

The purpose of requirement completeness in software engineering is to detect the situation when the results of the program do not match the input data. Even for the simplest of programs the number of cases of input data is astronomical.

Requirement Consistency

Consistency checks requirements in the document should not conflict or different description of the same

function.
An important aspect in the software development process is the consistency between various parts of the software system being designed and implemented. During the development of a system we are aware of the consistency problems and we usually solve these by special arrangements developed as part of the development of the software system.

Example of Consistency Requirement

An example of consistency is a sauce that is easy to pour from a pitcher. An example of consistency is when all tests that students take are graded using the same grading scale.

* Software Design Levels :-

1- Software architecture design level - The architecture design is the highest abstract version of the system. It identify the software as a system with many components interacting with each other. At this level the designers gets the idea of purposed solution domain.

2- High level design - The high level design break the 'single-entity multiple component' concept of architecture design into less abstracted view of sub-system and modules and depicts their interaction with each other. High level design focus on how the system along with all of its components can be implemented in form of module. It recognise modular structure of each subsystem and their relation and interaction among each other.

3- Detailed design - Detailed design deals with the implementation part of what is seen as a system and its sub system in the previous ~~to~~ two design.

It is more detailed towards modulars and implementations. It define logical structure of each module and their interface to communicate with other modules.

* Object Oriented Design :-
Object oriented design works around the entities and their characteristics instead of functions involve in the software system. This design strategies focus on entities and its characteristics. The whole concept of software solution revolves around the engaged entities.
Important concepts of object oriented design -

1- Objects - All entities involve in the solution design are known as objects. For example - person, bank, company and customers are read as an object, associated to it and has some methods to perform on the attributes.

2- Class - A class is a generalization description of an object. An object is an instance of a class. Class define all the attributes which are object can have

and methods which define the functionality of the objects. In the solution of design attributes are stored as a variable and functionality are define by means of method procedures.

3- Inheritance — Object oriented design allow similar classes to stack-up in hierarichal manner where the lower or subclasses can import, implement and reuse allow variables and methods from their immediate super-classes. These properties of object oriented design is known as inheritance. This make it easier to define specific class and to create generalize classes from specific one.

4- Polymorphism — Object oriented design languages provides a mechanism where methods performes similar tasks but vary in arguments can be assign same name. This is called polymorphism which allow a single interface performing task for different type. Depending upon how the function is invoke, respective portion of the code gets executed.

DATE: 16 / Feb / 2019

* Software Quality Requirement :-
Software quality requirement are specification of the quality of product, service, process or environment. Quality is any element, tangible or intangible that give things value beyond their functionality and features.

1- Availability - Is it available when and where I need to use it?
2- Installability - How easy is it to correctly install the product?
3- Integrity - Does it protect against unauthorize access and data loss?
4- Interoperability - How easily does it interconnect with other system?
5- Performance - How fast does it response or execute?
6- Recoverability - How quickly can the user recover from a failure?
7- Reliability - How long does it run before experiencing a failure?
8- Robustness - How will does it response to unaccepted operations?
9- Safety - How will does it protect against injure or damage?
10- Usability - How easy it for people to learn and use?

DATE: / /

11- Efficiency — How will does it utilize processor capacity, disk space, memory and other resource?

12- Flexibility — How easily can it be update with new functionality?

13- Maintainability — How easy it to correct defect or may change?

14- Portability — How easily can it be made to work on other platforms?

15- Scalability — How easily can it add more users, servers or other extensions?

16- Reusability — How easily can it be use components in other system?

17- Supportability — How easily will it be to support after installation?

18- Testability — Can I verify that it was implemented correctly?

* **Relationship between design and implementation**

An implementation is the process of converting the design as a program. Software design and implementation is the stage of the software engineering process at which an executable software system is developed. Software design and implementation activities are invariably enterlevel.

* Software design concept –

1. Abstraction
2. Refinement
3. Modularity
4. Software architecture
5. Control hierarchy
6. Software procedure
7. Structure partition
8. Data structure
9. Information hiding

1- Abstraction – Abstraction is the process of generalising by reducing the information content of a concept. It is an act of representing essential features without including the background details and explaination.

2- Refinement – It is the process of elaboration one or several instructions of a given program are decomposed into more detailed instructions. Abstraction and refinement are complementry concepts.

3- Modularity – Software architecture is divided into components called modules.

4- Software architecture – The overall structure of the software provides conceptual integrity for a system.

5- Control hierarchy – A program structure

that represent the organisation of a program component and implies a hierarchy of control.

6- Software procedure — It focus on the processing of each module individually.

7- Structure Partition — The program structure can be divided both horizontal and vertically.
Horizontal partition define seperate branches of modular hierarchy for each major program function.
Vertical partitioning suggests that control and work should be distributed top-down in the program structure.

8- Data structure — It is a representation of the logical structure among individual elements of data.

9- Information hiding — Modules should be specified and design so that information contain within a module is inaccessible to other modules that have no need for such information.