CHAPTER: 2

SOFTWARE REQUIREMENT ENGINEERING

(14 Marks)

2.1 Software Engineering Practices and Its importance, Core principles.
2.2 Communication Practices, Planning Practices, Modelling Practices, Construction Practices & Deployment Practices.
2.3 Requirement engineering: Requirement gathering & analysis, Types of requirements.
2.4 Software Requirement Specification: Need of SRS, Format, & It characteristics.

2.1 Software Engineering Practices and Its importance

- Software Engg. is systematic process of developing software.
- Practice is collection of concepts, principles, methods, and tools that a software engineer calls upon on a daily basis.
- Software Engg. Practice can be defined as set or collection of concepts, principles, methods, & tools that are used by Software practitioners on day to day basis for developing software product.
- Software engineer & managers apply the practice of Software Engg.
- Provides necessary technical and management how to be in getting the job done.

2.1.1 Seven Core Principles for Software Engg.

1) **The reason at all exists**
    - The software can be called as complete software if it satisfies all the requirements of customer.
    - Before beginning a software project is sure that the software has a business purpose.
2) **Keep it simple, stupid (KISS)/keep it simple, but perfect**
    - All design and implementation should be as simple as possible
    - Difficult design of system becomes very difficult to maintain & understand.
3) **Maintain the vision of the project**
    - A clear vision is essential to the project's success.
    - if the architectural vision of software system is not perfect then it will break a good designed system.
4) **What you produce, Others will consume**
    - Always specify, design, and implement knowing that someone else will later have to understand and modify what you did.
    - Project should be transparent so that the persons who work on it understand what has done.

- Other members in team may use the software which is an analyzed, designed, & constructed by other persons in a team.

5) Be open to the future

- Hardware technology system changes the software has to change for the purpose compatibility.
- Now a day's software lifetime is measured in months & not in years due to outdating problem of software product as well as languages.

6) Plan ahead for software reuse

- Reuse of software reduces the long-term cost and increases the value of the program and the reusable components.
- The reusable software can be used in the development of other applications even.(e.g. C++ lang. possible reusable.)

7) Think, then act

- When we think about something, we get more ideas.
- Placing clear, complete thought before action will almost always produce better results

2.2 Communication Practices:-

Principle 1 "Listen Carefully"

- To understand customers' requirements carefully & perfectly. It is necessary to become good listener.
- Listen to the speaker and concentrate on what is being said.

Principle 2 "Be prepared before you communicate"

- It is necessary to prepare the agenda for the meeting or discussion.
- Prepare before you meet by researching and understanding the problem.

Principle 3 "Be prepared to facilitate the activity"

- Every communication meeting should have a leader (a facilitator) to keep the conversation moving in a productive direction, to mediate any conflict that does occur, and to ensure than other principles are followed.

Principle 4 "Face-to-face communication is best"

- But also have a document or presentation to focus the discussion
- But it usually works better when some other representation of the relevant information is present.

Principle 5. ***Take notes and document decisions***

- Someone participating in the communication should serve as a "recorder" and write down all important points and decisions.

Principle 6. ***Strive for collaboration.***

- Collaboration and Consensus occur.

- When the collective knowledge of members of the team is used to describe product or system functions or features.

Principle 7. ***Stay focused; modularize your discussion.***

- Stay focused on a topic; modularize your discussion
- The more people involved in any communication, the more likely that discussion will bounce from one topic to the next.

Principle 8. ***If something is unclear, draw a picture.***

Principle 9. ***Keep the discussion to move on.***

Principle 10. ***Negotiation is successful when both parties agree.***

2.2.1 Planning Practices:-

Principle 1. ***Understand the scope of the project.***

- If we don't know exact destination to reach, then meaningless to travel.
- Thus s/w planning provides the exact goal for s/w development team.
- Scope provides the software team with a destination.

Principle 2. ***Involve stakeholders in the planning activity.***

- Software planning includes complete estimation.
- In each & every activity customer should be involved while developing project.

Principle 3. ***Recognize that planning is iterative.***

- In incremental model of s/w process customer may suggest modifications after every delivery of increment. Accordingly the planning must be changed.
- Thus plan changes time to time.

Principle 4. ***Estimate based on what you know.***

- The intent of estimation is to provide an indication of effort, cost, and task duration, based on the team's current understanding of the work to be done.
- If information is vague or unreliable, estimates will be equally unreliable.

Principle 5. ***Consider risk as you define the plan.***

- If you have identified risks that have high impact and high probability, contingency planning is necessary.
- In addition, the project plan (including the schedule) should be adjusted to accommodate the likelihood that one or more of these risks will occur.

Principle 6. ***Be realistic. (Try to deal with practical way)***

- Project should be realistic all practical aspects have to be considered.
- As human beings are involved in project there are chances for delays, mistakes, wrong decisions etc.

Principle 7. ***Adjust granularity as you define the plan.***

- *"Granularity"* refers to the "level of detail" for particular specification.

- Adjust granularity in the project plan.

Principle 8. ***Define how you are going to achieve quality.***

- The formal technical reviews are conducted to assure the quality.
- FTR is meeting conducted by technical staff.
- FTR not problem solving activity but it is applied to find out defects in any stage of s/w developer.

Principle 9. ***Describe how you aim to accommodate change.***

- If change is requested in early stages of s/w development it can be easily carried out. But if change is requested in later stages, cost of change rapidly increases.

Principle 10. A*lways keep track of the plan & make changes as required.*

- Project plan should be tracked to view the progress on day to day basis.

2.2.2 Modelling Practices:-

- Models are created for better understanding of the actual entity to be built or design.
- Entity is a physical thing. E.g. building, plane, machine etc.
- Software Engg. Work classes of models are created analysis & design models.
- Analysis Modeling
- It represents 3 different domains
  1. Information domain
  2. Functional domain
  3. Behavioral domain
- Design Modeling
- It represents 3 different domains
  1. The architecture
  2. The user interface
  3. Components level details

2.2.2.1 Analysis Modeling Principles

- Analysis represents the customer requirements. e. g. analysis steps include algorithms specifications of the program or process.

Principle 1. ***The information domain of a must be represented and understood***

- It uses "data flow diagram(DFD)" to show such information domain
- Information flow includes following details:-
  1. Input flow into system
  2. Output flow from the system.
  3. Data collection by data stores to collect data in the system.

Principle 2. ***The functions of the software must be defined clearly***

- Algorithms provide function details.

- Functions are the processes those transform the input flow to output flow.

Principle 3. ***The behavior of the software or system must be defined clearly.***

- Analysis model uses "state transition diagrams (STD)" to represent the behavior of the s/m clearly.
- STD shows how the system makes transition from one state to another state.

Principle 4. ***Clear hierarchy among information, functions, & behavior must be shown.***

- Information, functions, & behavior of system must be represented using proper hierarchy. i.e. levels or layers.

Principle 5. ***The analysis task should be clear to convert it into design model***

- If analysis of requirements is clear & simple then it will be easy for design.

2.2.2.2 Design Modeling Principles

- Design model is equivalent to the architectural plan of house.
- It provides concrete specification for the construction of the software.
- Design steps include the design of flowchart for the pictorial reprehension of requirement.
- The deign models is afterwards converted into coding using appropriate programming language.

Principle 1. ***The design should be traceable to the analysis model***

- Using elements of analysis model, design model is constructed.
- E.g. from information domain at analysis model, the architectural design is developed.

Principle 2. ***Always consider the software architecture of the system to be built.***

- Data flow architecture design: - Input data flow after processing generates output data.

Principle 3. ***Design of data is as important as design of processing functions***

- Design model includes the entity- relationship diagrams to show the relationship among different data objects.

Principle 4. ***Interfaces (both internal and external) must be designed with care***

- Data may flow from one component to another component internal & external interface must be designed properly.

Principle 5. ***User interface design must be satisfy all need of end user.***

- User interface should be as simple as possible.

Principle 6. ***Component-level design should be functionally independent***

- Functions should be designed such that all tasks of single software component.

Principle 7. ***Components should be loosely coupled to one another and to the external environment***

- For good design interconnection among different components should be minimum.(coupling means interconnection among different modules)

Principle 8. ***Design representations (models) should be easily understandable***

- Design should be as simple as possible.

Principle 9. ***The design should be developed iteratively***

- In case of incremental process model customer may suggest modifications after such delivery of the increment.

2.2.3 Construction Practices:-

- The construction activity includes a set of coding and testing tasks that lead to operational software that is ready for delivery to the customer or end user.

Coding:-

- Design details are implemented using appropriate programming language.
- Using suitable programming language to generate the source code. E.g. C, C++, VB, Java, .NET, PHP etc.
- Using automated tools like Microsoft front page, Dreamweaver where code is automatically generated.

Testing:-

- To check whether flow of coding is correct.
- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as-yet undiscovered error.

2.2.3. 1. Coding Principles (Preparation before coding)

1) Understand the problem you are trying to solve.
2) Understand basic design principles and concepts.
3) Select appropriate a programming language to implement solution.
4) Select a programming environment that will suitable for writing the code.
5) Create a set of unit tests that will be applied once the component you code is completed.

2. Actual Coding Principles (As you begin coding)

1) The algorithms you are using must follow structured programming principles.
2) Use suitable data structures those will fit for coding.
3) Software architecture and interfaces are to be implemented as per design specifications.
4) Keep minimum nested conditions.
5) Keep minimum nested loops.
6) Select meaningful variable names and follow other local coding standards.
7) Write code that is self-documenting i.e. provide help comments for all steps.
8) Create a visual layout (e.g., indentation and blank lines) that aids code understanding.

3. Coding (Validation) Principles (After completing the first round of code)

1) Conduct a code walkthrough
2) Perform unit tests (black-box and white-box) and correct errors you have uncovered
3) Refactor the code.

2.2.3.2 Testing Principles

- Software testing is an activity, that ensures:
  1. Quality of the software.
  2. Customer satisfaction.
  3. Removal of errors.
  4. Verification & validation of s/w product.
  5. Compatibility of the software.

Principle 1. ***Tests must be conducted to validate customer requirements.***

- Basic aim of testing is to find defects from the developed modules.
- Defects are considered from customer's point of view.

Principle 2. ***Tests should be well planned before starting testing work***

- Testing plan can be started as soon as analysis model is completed.
- All tests can be planned and designed before any code has been generated.

Principle 3. ***The Pareto principle applies to software testing.***

- According to this principle, of 80-20% rule 80% of all errors found in testing will likely be traceable to 20% of all program modules.

Principle 4. ***Testing should begin "in the small" and progress toward testing "in the large."***

- Unit testing--> integration testing--> validation testing--> system testing

Principle 5. ***Accept that testing of every combination of pats is not possible.***

- Each s/w module as unit contains too many conditions & nested conditions.
- Hence it is not possible to check out coming results satisfy every combination of inputs.
- E.g. not possible to check every if-else combination of s/m but tests should be conducted all logical paths will be traced at least once.

2.2.4 Deployment Practices:-

- The deployment activity encompasses three actions: delivery, support, and feedback.
- Because modern software process models are evolutionary or incremental in nature, deployment happens not once, but a number of times as software move toward completion.
- Each delivery cycle provides the customer and end users with an operational software increment that provides usable functions and features.
- Each support cycle provides documentation and human assistance for all functions and features introduced during all deployment cycles to date.
- Each feedback cycle provides the software team with important guidance that results in modifications to the functions, features, and approach taken for the next increment.

Principle 1. ***Customer expectations for the software must be managed.***

- At time of s/w delivery developer must have skill to manage the customer expectations.

Principle 2. ***A complete delivery package should be assembled and tested.***

- All executable software, support data files, support documents, and other relevant information should be assembled and thoroughly beta-tested with actual users.

Principle 3. ***Record keeping mechanism must be established for customer support.***

- Customer support is important factor in deployment phase if proper support is not provided, customer will not be satisfied.

Principle 4. ***Appropriate instructional materials must be provided to end users.***

- Actual project delivery includes all the documentations, help file & guidance for handling the s/w by user.

Principle 5. ***Buggy software should be fixed first, delivered later.***

- Under time pressure, some software organizations deliver low-quality increments with a warning to the customer that bugs "will be fixed in the next release."
- Don't deliver any defective or buggy s/w to the customer.

2.3 Requirement engineering Types of requirements.

2.3.1 Requirement gathering & analysis

Q. Enlist requirement Gathering and Analysis for web based project for registering candidates for contest. S-19

- Requirement gathering includes suggestions and ideas for ways to best capture the different types of requirement (functional, system, technical, etc.) during the gathering process.

1. Functional requirements

- The functional requirements are the requirements that will enable solving the real world problem. The web based project must be able to register the candidates for contest.

2. Non-functional requirements

- These requirements aim at providing support, security and facilitate user interaction segment of the website.
- The project must enable the candidates to safely enter their passwords and other biometric information.
- There must be no repetition in registration of candidates i.e the candidates must be registered only once.

3. **Business requirements**

- They are high-level requirements that are taken from the business case from the projects.
- For eg:-

Qualifying criteria	Allowed/Disallowed
Indian Nationality Registration	Allowed
Age>18	Allowed
No criminal record	Allowed

4. **Architectural and Design requirements**:

- These requirements are more detailed than business requirements. It determines the overall design required to implement the business requirement.
- The web based project must be supported by different operating systems , PC and mobile compatibility etc.
- The hardware must be integrated so as to accept the fingerprint details of a candidate and register him in the system.
- The database of the project must be updated.

5. **System and Integration requirements**:

- At the lowest level, we have system and integration requirements.
- It is detailed description of each and every requirement.
- It can be in form of user stories which is really describing everyday business language.
- The requirements are in abundant details so that developers can begin coding.

6. Documenting the requirement using traceability matrix

- A Traceability Matrix is a document that co-relates any two base line documents that require a many-to-many relationship to check the completeness of the relationship.
- It is used to track the requirements and to check the current project requirements are met.

Req no	Description	Test case ID	Status
1	Login	TC1	TC1 Pass
2	Feed in biometric details	TC2	TC2 Pass

2.3.2 Requirement engineering & Types of requirements.

Q. Explain six function of requirement engineering process. W-19

- The broad spectrum of tasks and techniques that lead to an understanding of requirements is called requirements engineering.
- It starts during the communication activity and continues into the modeling activity.
- Requirements engineering provides the appropriate mechanism for understanding what the customer wants by analyzing need, assessing feasibility negotiating a reasonable solution, specifying the solution ambiguously, validating the specification, and managing the requirements as they are transformed into an operational system.
- **It encompasses seven distinct tasks:** inception, elicitation, elaboration, negotiation, specification, validation, and management.

- **Need of RE**
  - Core process of RE is "**need analysis problem definition, goal identification, requirement definition, solution expectations, enabling the s/w engineer to determine the s/w requirement specification (SRS) to solve the customers problems**".
  - Designing & building the computer s/w will not satisfy customer's requirements. If requirements analysis is wrong, then design will be wrong, ultimately coding will be wrong.
  - Finally s/w expectations will not match with outcomes.
  - Hence requirement engineering (RE) should be carried out carefully.

- **Requirements Engineering Tasks**
  - **Seven distinct tasks**
    - **Inception**
    - **Elicitation**
    - **Elaboration**
    - **Negotiation**
    - **Specification**
    - **Validation**
    - **Requirements Management**

1. Inception

- Inception means beginning.
- It is usually said that "well beginning is half done" but it is always problematic for the developer that, "from where to start".
- RE itself is a "communication intensive activity".
- Customer & developer meet & they decide the overall scope & nature of problem statement.
- The aim is: - during inception, the requirements engineer asks a set of questions to establish…
  - A basic understanding of the problem
  - Who will use the s/w?
  - The people who want a solution
  - The nature of the solution that is desired
  - The effectiveness of preliminary communication and collaboration between the customer and the developer

2. Elicitation

- "To draw out the truth or reply from anybody"
- Elicitation is a task that helps the customer to define what is required.
- Eliciting requirements is difficult because of

- Problems of scope in identifying the boundaries of the system or specifying too much technical detail rather than overall system objectives
- Problems of understanding what is wanted, what the problem domain is, and what the computing environment can handle (Information that is believed to be "obvious" is often omitted)
- Problems of volatility (means change from one state to another) because the requirements change over time.

3. Elaboration

- It means "To work out in detail".
- During elaboration, the software engineer takes the information obtained during inception and elicitation and begins to expand and refine it.
- Elaboration action is called as "analysis modeling" action.
- Elaboration focuses on developing a refined technical model of software functions, features, and constraints
- It is an analysis modeling task
  - Use cases are developed
  - Domain classes are identified along with their attributes and relationships.
  - State machine diagrams are used to capture the life on an object.
- The end result is an analysis model that defines the functional, informational, and behavioral domains of the problem.

4. Negotiation

- It means "discussion on financial & other commercial issues".
- During negotiation, the software engineer reconciles the conflicts between what the customer wants and what can be achieved given limited business resources.
- Requirements are ranked (i.e., prioritized) by the customers, users, and other stakeholders.

5. Specification

- A specification is the final work product produced by the requirements engineer
- It is normally in the form of a software requirements specification (SRS).
- It formalizes the informational, functional, and behavioral requirements of the proposed software in both a graphical and textual format.

6. Validation

- All previous work completed will be just useless & meaningless if it is not validated against the customer expectations.
- During validation, the work products produced as a result of requirements engineering are assessed for quality.
- The specification is examined to ensure that
  - All software requirements have been stated unambiguously

- Inconsistencies, omissions, and errors have been detected and corrected
- The work products conform to the standards established for the process, the project, and the product

7. Requirements Management

- During requirements management, the project team performs a set of activities to identify, control, and track requirements and changes to the requirements at any time as the project proceeds
- Each requirement is assigned a unique identifier
- The requirements are then placed into one or more traceability tables.
- These tables may be stored in a database that relate features, sources, dependencies, subsystems, and interfaces to the requirements

2.4 Software Requirement Specification

Q. Define software requirement specification. W-19

Concept:

- SRS are generated at the end.
- It is document that completely describes what proposal s/w should do without describing how will do it.
- SRS is also helping the client to understand their own needs.
- IEEE defines SRS as a document that clearly & precisely describes each of the essential requirements of the s/w & external interface.
- The main goal of SRS is completely & consistently specifies the technical requirements for s/w product.
- **SRS plays a significant role as it contains following importance in detail:**
    1. Complete or total information description.
    2. A full functional description.
    3. The representation of system behavior.
    4. Indication or sign or performance requirements & design constraint.
    5. Suitable validation criteria
    6. Other information related to requirements.

Need of SRS

Q. State need of software requirement specification (SRS). S-19

The need of SRS document is to provide

- A detailed overview of software product, its parameters and goals.
- The description regarding the project's target audience and its user interface hardware and software requirements.
- How client, team and audience see the product and its functionality.

General format of SRS

1. Information
2. Information description
3. Functional description
4. Behavioral description
5. Validation criteria
6. Bibliography
7. Appendix

----------End of Chapter----------