2) Software Engineering Practices & Software Requirements Engineering

Software engineering practice is a broad array (collection) of principles, concepts, methods, approaches and tools that we must ~~custom~~ consider as software is planned and developed.

Importance of SE practices:-

① The software process provides a road map to everyone involved in the creation of software for getting to successful destination.

② Practices provide us with details need to drive along the road. They tells us where the bridges, the roadblocks, and the forks are located

③ Practices help us to understand the concepts and principles that must be understood and followed to drive safely and rapidly.

④ Practices instruct us about how to drive, where to slow down and where to speed up.

Software Engineering practices - The Essense of Practice:

① Understand the problem (communication & analysis)
② Plan a solution (Modelling & software design)
③ Carry out the plan (Code generation)
④ Examine the result for accuracy (testing and quality assurance)

① Understand the problem:
- Q) Who are the stakeholders?
- Q) What data, functions, and features are required to properly solve the problem?
- Q) Is it possible to represent smaller problems that may be easier to understand?
- Q) Can a problem be represented geographically?

② Plan the solution:
- Q) Have you seen similar problem before?
- Q) Has a similar problem has been solved?
- Q) Can sub-problems be defined?
- Q) Can you represent a solution in a manner that leads to effective implementation.

③ Carry out the plan:
- Q) Does the solution confirm to the plan?
- Q) Is each component part of the solution provably correct?

④ Examine the result:
- Q) Is it possible to test each component part of the solution
- Q) Does the solution produce results that confirm to the data, functions and the features that are required?

Core principles of Software Engineering:

The word principle is defined as "an important underlying law or assumption required in a system of thought."
David Hooker has proposed seven core principles that focus on software engineering practices as a whole. (Explain the principles in one sentence. Here, they are only listed).

Principle 1: The Reason it all exists
Principle 2: KISS (keep it simple, stupid)
Principle 3: Maintain the vision
Principle 4: What you produce, others will consume
Principle 5: Be open to the future.
Principle 6: Plan Ahead for Reuse.
Principle 7: Think!

Communication principles:

Principle 1: Listen
Principle 2: Prepare before you communicates
Principle 3: Someone should facilitate the activity
Principle 4: Face to face communication is best
Principle 5: Take notes and document decision
Principle 6: Strive for collaboration
Principle 7: Stay focused; modularize your decision
Principle 8: If something is unclear, draw a picture
Principle 9: Once you agree to something, move on
Principle 10: Negotiation is not a contest or a game. It works best when both parties win.

Page No.
Date

Planning Principles:

Principle 1 : Understand the scope of the project.
Principle 2 : Involve the stakeholders in the planning activity.
Principle 3 : Recognize the planning is iterative.
Principle 4 : Estimate based on what you know.
Principle 5 : Consider risk as you define the plan.
Principle 6 : Be realistic.
Principle 7 : Adjust granularity as you define the plan.
Principle 8 : Define how you intend to ensure quality.
Principle 9 : Describe how you intend to accomodate change.
Principle 10 : Track the plan frequently and make adjustments as required.

Modelling practices:

In software engineering work, two class of models can be created:

1. Analysis model :- It represents customer requirements by depicting the software in three different domains:
   * Information domain
   * Functional domain
   * Behavioural domain
2. Design model :- It represents characteristics of the software that help practitioners to construct it effectively:
   * Architecture
   * User interface
   * Component level details.

Analysis modelling principles:

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

Principle 2: The functions that the software performs must be defined.

Principle 3: The behaviour of the software (as a consequence of external events) must be represented.

Principle 4: The models that depict information function and behaviour must be partitioned in a manner that uncovers detail in a layered (hierarchical) fashion.

Principle 5: The analysis task should move from essential information toward implementation detail.

Design modelling principles:

Principle 1: Design should be traceable to the analysis model.

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

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

Principle 4: Interfaces (both internal & external) must be designed with care.

Principle 5: User interface design should be tuned to the needs of the end user. However in every case, it should stress ease of use.

Principle 6: Component level design should be functionally independent.

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

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

Principle 9: The design should be developed iteratively. With each iteration, the designer should strive for greater simplicity.

Construction practices:

Coding Principles and concepts:

- Preparation principles: Before you write the one line of code

Principle 1: Understand the problem you are trying to solve

Principle 2: Understand basic design principles and concepts

Principle 3: Select a programming language that meets the need of software to be built and the environment in which it will operate.

Principle 4: Select a programming environment that provides tools that will make your work easier.

Principle 5: Create a set of unit tests that will be applied once the component is completed

- Programming principles: As you begin writing the code

Principle 1: Constrain our algorithms by ~~followed~~ following structured programming practice.

Principle 2: Consider the use of pair programming.

Principle 3: Understand the software architecture and create interfaces that are consistent with it.

Principle 4: Keep conditional logic as simple as possible

Principle 5: Create nested loops in easily testable manner.

Principle 6: Select meaningful variable names and follow other local coding standard. Write self documenting code.

Principle 7: Create a visual layout that aids understanding.

Date

- Validation principles: After you have completed your first coding pass

Principle 1: Conduct a code walkthrough when appropriate

Principle 2: Perform unit test and correct errors you have uncovered.

Principle 3: Refractor the code.

Testing practices

Principle 1: All tests should be traceable to customer requirements

Principle 2: Tests should be planned long before testing begins

Principle 3: The Pareto principle applies to software testing.

Principle 4: Testing should begin "in the small" and progress towards testing "in the large"

Principle 5: Exhaustive testing is not possible.

Deployment practices :-

Principle 1: Customer expectation for the software must be managed

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

Principle 3: A support regime must be established before the software is delivered.

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

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

Concept of Requirement Engineering:

Requirement engineering is a set of task that helps in understanding the following:

1) Business impact of software
2) Exact expectations of customer
3) Interaction of user with system.

Need of Requirement Engineering:

- Designing and building the elegant computer software will not satisfy customer's requirements. If requirement analysis is wrong, then design will be wrong.
- Ultimately coding will be wrong. Finally, expectations from the software will not match with outcomes. Hence requirement engineering should be carried out carefully.

Requirement Engineering tasks:

① Inception
② Elicitation
③ Elaboration
④ Negotiation
⑤ Specification
⑥ Validation
⑦ Requirement management

① Inception:- Inception means beginning. It is usually said that well begun is half done. But it is always problem for the developer from where to start.

Project inception software developers will ask following set of questions:

- Who is behind the request for this work?
- Who will use the solution?
- What will be the economic benefits of a sucessful solution?

At project inception, we establish a basic understanding of a problem, the people who want the solution, and the effectiveness of preliminary communication and collaboration between other stakeholders and the software team.

(2) Elicitation :- Elicitation means to draw out the truth from anybody. In requirement engineering, elicitation is a task that helps the customer to define what is required?

There are numbers of problems encountered during elicitation:

- Problem of scope :- Many a times customer states unnecessary technical detail. These unnecessary details may confuse the developer rather than clarify, overall system objectives.
- Problem of understanding :- Sometimes both customer as well as developer have poor understanding of:
  * What is needed?
  * Capabilities & limitations of the computing environment.
  * Understanding of problem domain
  * Specify requirements those conflict with other customers and users.

- Problem of volatility :- The requirements change over the period of time. This is also a major problem while deceiding fixed set of requirements.

③ Elaboration :- Elaboration means to work out in detail. The information that is received in the beginning during inception and elicitation phases is expanded and medications are made during elaboration.
The result of elaboration is an analysis model which defines the informational, behavioural and functional domain of the problem.

④ Negotiation :- Negotiation is discussion on financial and commercial issues. In this step customer, user and other stakeholders discuss to decode :
* To rank the requirements
* To decide priorities
* To decide risks
* To finalize project costs
* The impact of above issues on project, cost and delivery rate.

⑤ Specification:- The specification is the final work product produced by requirement engineer. The specification may be:

① A written document
② A set of graphical model
③ A formal mathematical model
④ A collection of scenarios
⑤ A prototype
⑥ Any combination of above

⑥ Validation:- The work products produced during requirements engineering are accessed for quality during a validation step.
Requirements validation examines the specification to ensure that:

* All software requirements stated unambiguously.
* Inconsistencies, omissions and errors have been detected and corrected.
* The work products to confirm the standards.

⑦ Requirement Management:- Requirement management is a set of activities that helps the project to identify, control and track requirements and changes. Once the requirements are identified traceability tables can be developed.

- Features traceability table
- Source traceability table
- Dependency traceability table
- Subsystem traceability table
- Interface traceability table

Software Requirement specification (SRS):

Software Requirement specification is a document that specifies:

① Complete or total information description
② A full functional description.
③ The representation of system behaviour.
④ Indication or sign of performance requirements and design constraints.
⑤ Suitable validation criteria
⑥ Other information related to requirements.

General format of SRS:

① Introduction:- This an introduction to software requirement specification. It can be used to define the Purpose, Document conventions, Intended Audience and Reading suggestions, Project scope and references. *

② Overall descriptions:- This section contains overall description of product. It defines Product perspective, features, User documentation, Operating environment, etc.

③ System features:- This template illustrates organizing the functional requirements for the product by system features, the major services produced by the product.

④ External Interface Requirements:- It defines user interfaces, Hardware and software interfaces and communication interface that can be used.

⑤ Other non-functional requirements :- This section defines Performance requirements, safety requirements, security requirements and software Quality Attributes.

⑥ Appendices :- The appendix section contain following information

* Tabular data
* Full descriptions of algorithms
* Charts
* Graphs
* Other relevant material