PAPER NO-XII

SOFTWARE ENGINEERING

BSC-THIRD YEAR

PRESENTED BY - PROF. KUTWAD A.G.

Introduction to requirement engineering

- Requirements engineering provides the appropriate mechanism for understanding what the customer wants, analyzing need, assessing easibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification.
-
- Requirement engineering consists of seven different tasks as follow:

  1. Inception
-
- Inception is a task where the requirement engineering asks a set of questions to establish a software process. In this task, it understands the problem and evaluates with the proper solution. It collaborates with the relationship between the customer and the developer. The developer and customer decide the overall scope and the nature of the question.
-
- **2. Elicitation**
-

  Elicitation means to find the requirements from anybody. The requirements are difficult because the **following problems occur in elicitation**.

  Problem of scope: The customer give the unnecessary technical detail rather than clarity of the overall system objective.
  Problem of understanding: Poor understanding between the customer and the developer regarding various aspect of the project like capability, limitation of the computing environment.
  Problem of volatility: In this problem, the requirements change from time to time and it is difficult while developing the project.

3. Elaboration

In this task, the information taken from user during inception and elaboration and are expanded and refined in elaboration.Its main task is developing pure model of software using functions, feature and constraints of a software.

4. Negotiation

In negotiation task, a software engineer decides the how will the project be achieved with limited business resources.To create rough guesses of development and access the impact of the requirement on the project cost and delivery time.

5. Specification

In this task, the requirement engineer constructs a final work product. The work product is in the form of software requirement specification. In this task, formalize the requirement of the proposed software such as informative, functional and behavioral. The requirement are formalize in both graphical and textual formats.

6. Validation

The work product is built as an output of the requirement engineering and that is accessed for the quality through a validation step.The formal technical reviews from the software engineer, customer and other stakeholders helps for the primary requirements validation mechanism.

7. Requirement management

It is a set of activities that help the project team to identify, control and track the requirements and changes can be made to the requirements at anytime of the ongoing project .These tasks start with the identification and assign a unique identifier to each of the requirement.After finalizing the requirement traceability table is developed. The examples of traceability table are the features, sources, dependencies, subsystems and interface of the requirement.

THE DESIGN PROCESS

- Software design is an iterative process through which requirements are translated into a “blueprint” for constructing the software.

- The design is represented a high level of abstraction— a level that can be directly traced to the specific system objective and more detailed data, functional, and behavioral requirements.

Software Quality Guidelines and Attributes

- Throughout the design process, the quality of the evolving design is assessed. Three characteristics that serve as a guide for the evaluation of a good design: Each of these characteristics is actually a goal of the design process.
- The design must implement all of the explicit requirements.
- The design must be a readable, understandable guide for those who generate code and for those who test.
- The design should provide a complete picture of the software, addressing the data, function and behavior.

Quality Guidelines

- A design should exhibit an architecture that has been created using recognizable architectural styles or patterns.
- A design should be modular.
- A design should contain distinct representations of data, architecture, interfaces, and component.
- A design should lead to components that exhibit independent functional characteristics.
- A design should lead to interfaces that reduce the complexity of connections between components and with the external environment.
- A design should be represented using a notation that effectively communicates its meaning.
- A design should be derived using a repeatable method that is driven by information obtained during software requirements analysis.

Quality Attributes

Quality attributes represent a target for all software design:

- Functionality - It is assessed by evaluating the feature set and capabilities of the program.
- Usability – It is assessed by considering human factors
- Reliability - It is evaluated by measuring the frequency and severity of failure, the accuracy of output results, the mean-time-to-failure (MTTF), the ability to recover from failure, and the predictability of the program.
- Performance - It is measured by considering processing speed, response time, resource consumption, throughput, and efficiency.
- Supportability – It combines the ability to extend the program.

ELEMENTS OF SOFTWARE QUALITY ASSURANCE

Software quality assurance(SQA) encompasses a broad range of concerns(elements) and activities that focus on the management of software quality.

- **Standards** - The IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents. The job of SQA is to ensure that standards that have been adopted are followed and that all work products conform to them.

- **Reviews and audits** - Technical reviews are a quality control activity performed by software engineers, Their intent is to uncover errors. Audits are a type of review performed by SQA with the intent of ensuring that quality guidelines are being followed for software engineering work.

- **Testing** - Software testing primary goal is to find errors. The job of SQA is to ensure that testing is properly planned and efficiently conducted.

- **Error/defect collection and analysis** - SQA collects and analyzes error and defect data to better understand how errors are introduced and what software engineering activities are best suited to eliminating them.

- **Change management** - change is not properly managed, change can lead to confusion, and confusion almost always leads to poor quality. SQA ensures that adequate change management have been instituted.

- **Education** - Every software organization wants to improve its software engineering practices. A key contributor to improvement is education of software engineers, their managers, and other stakeholders. The SQA organization takes the lead in software process improvement and is a key proponent and sponsor of educational programs.

- **Security management** - SQA ensures that appropriate process and technology are used to achieve software security.

- **Safety** - SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk

- **Risk management** - SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

- **Vendor management** - The job of the SQA organization is to ensure that high-quality software results by suggesting specific quality practices that the vendor should follow, and incorporating quality mandates as part of any contract with an external vendor.

SOFTWARE TESTING FUNDAMENTALS

The goal of testing is to find errors, and a good test is one that has a high probability of finding an error.

- **Testability** - "Software testability is simply how easily can be tested." The following characteristics lead to testable software.

- Operability - "The better it works, the more efficiently it can be tested." If a system is designed and implemented with quality in mind, relatively few bugs will block the execution of tests, allowing testing to progress without fits and starts.

- Observability - "What you see is what you test." Inputs provided as part of testing produce distinct outputs. System states and variables are visible or queriable during execution. Incorrect output is easily identified. Internal errors are automatically detected and reported. Source code is accessible.

- Controllability - "The better we can control the software, the more the testing can be automated and optimized."

- Decomposability - "By controlling the scope of testing, we can more quickly isolate problems and perform smarter retesting." The software system is built from independent modules that can be tested independently.

- Simplicity - “The less there is to test, the more quickly we can test it.” The program should exhibit functional simplicity, structural simplicity and code simplicity .

- Stability - “The fewer the changes, the fewer the disruptions to testing.” Changes to the software are infrequent, controlled when they do occur, and do not invalidate existing tests. The software recovers well from failures.

- Understandability - “The more information we have, the smarter we will test.” The architectural design and the dependencies between internal, external, and shared components are well understood. Technical documentation is instantly accessible, well organized, specific and detailed, and accurate. Changes to the design are communicated to tester.

Test Characteristics

- A good test has a high probability of finding an error.
- A good test is not redundant.
- A good test should be “best of breed”.
- A good test should be neither too simple nor too complex.