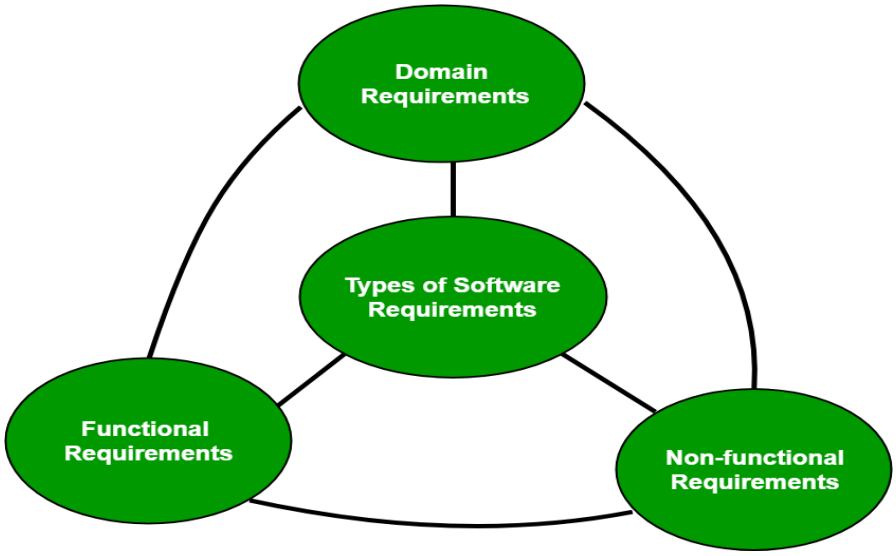
Software engineering unit 2

Software Engineering (Dr. A.P.J. Abdul Kalam Technical University)

Software Engineering (Unit-2)

What is a requirement?

• A Requirement can be defined as a condition or capability needed by a user to solve a problem or achieve an objective or A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.



- **Functional Requirements:** These are the requirements that the end user specifically demands as basic facilities that the system should offer. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

•Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. Nonfunctional requirements, not related to the system functionality, rather define how the system should perform.

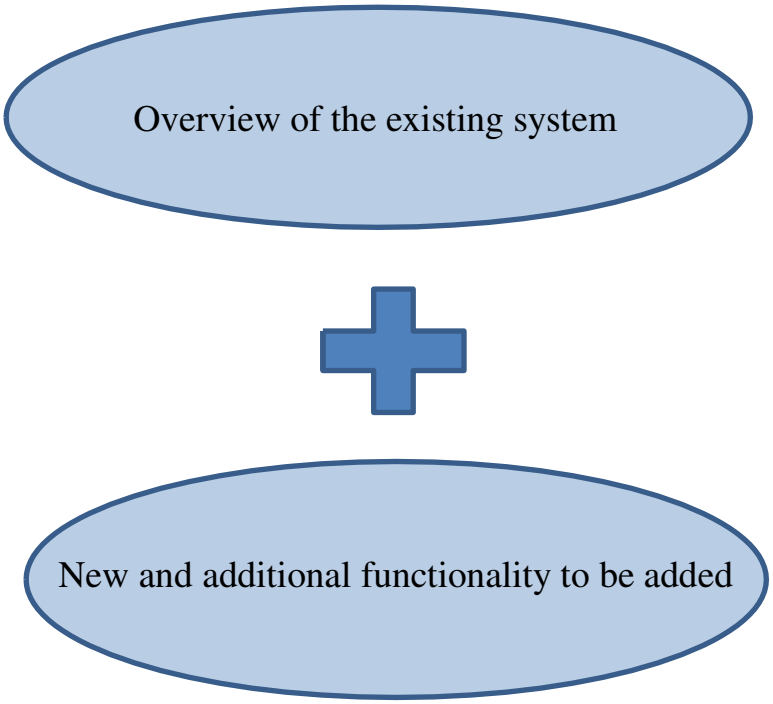
•Domain requirements: Domain requirements are the requirements which are characteristic of a particular category or domain of projects. For instance, in academic software that maintains records of a school or college, the functionality of being able to access the list of faculty and list of students of each grade is a domain requirement. These requirements are therefore identified from that domain model and are not user specific.

Functional Requirements	Non-functional requirements
Functional requirements help to understand the functions of the system.	They help to understand the system's performance.
Functional requirements are mandatory.	While non-functional requirements are not mandatory.
They are easy to define.	They are hard to define.
They describe what the product does.	They describe the working of product.
It concentrates on the user's requirement.	It concentrates on the expectation and experience of the user.
It helps us to verify the software's functionality.	It helps us to verify the software's performance.
These requirements are specified by the user.	These requirements are specified by the software developers, architects, and technical persons.
There is functional testing such as API testing, system, integration, etc.	There is non-functional testing such as usability, performance, stress, security, etc.
Examples of the functional requirements are - Authentication of a user on trying to log in to the system.	Examples of the non-functional requirements are - The background color of the screens should be light blue.
These requirements are important to system operation.	These are not always the important requirements, they may be desirable.
Completion of Functional requirements allows the system to perform, irrespective of meeting the non-functional requirements.	While system will not work only with non-functional requirements.

Requirement Engineering

- **Requirements engineering (RE)** refers to the process of defining, documenting, and maintaining requirements in the engineering design process.
- Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system.

- **Input to RE:**



Requirement Engineering Process

1) Feasibility Study
2) Requirement Elicitation and Analysis
3) Software Requirement Specification
4) Software Requirement Validation
5) Software Requirement Management

Feasibility Study

- A feasibility study specifies whether the proposed software project is practically possible or not. Key points to determine the importance of feasibility study are:

1. It prevents you from committing, investing your resources and capital for an impractical project.
2. It might discover new ideas that might completely change the scope of your software.
3. Improves the confidence of the team to concentrate on their project.
4. Validate the need for the software.
5. Estimates the risk involved in developing the software and analyzed if they can be minimized.

Types of Feasibility Study

1. **Technical Feasibility:-** The technical feasibility study determines whether the current technology and technical staff available with the organization will accomplish the software's development or not.
2. **Organizational Feasibility:** Though the software project is feasible from the technical aspect what if you need to make major organizational changes.
   - Identify whether the organizational staff is ready to accept the new software or not
   - Is the organizational staff technical sound to cope up with new technologies?
   - Does the organization hardware infrastructure capable to support the new software?

- **Competition:** Before we start developing the software we must consider the current trends of the market and the competitive software available in the market. This will help in the development of the software.

- **Uniqueness:** Though there are several competitor software available in the market the newly developed software must be known for its uniqueness. It must offer something that the existing software in the market doesn't offer.

- **Performance:** The performance of the software needs to be traced to whether it is performing financially and technically well corresponding to its requirement.

- **Economic feasibility**: It determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study.

- Cost incurred on software development to produce long-term gains for an organization.
- Cost required to conduct full software investigation.
- Cost of hardware, software, development team, and training.

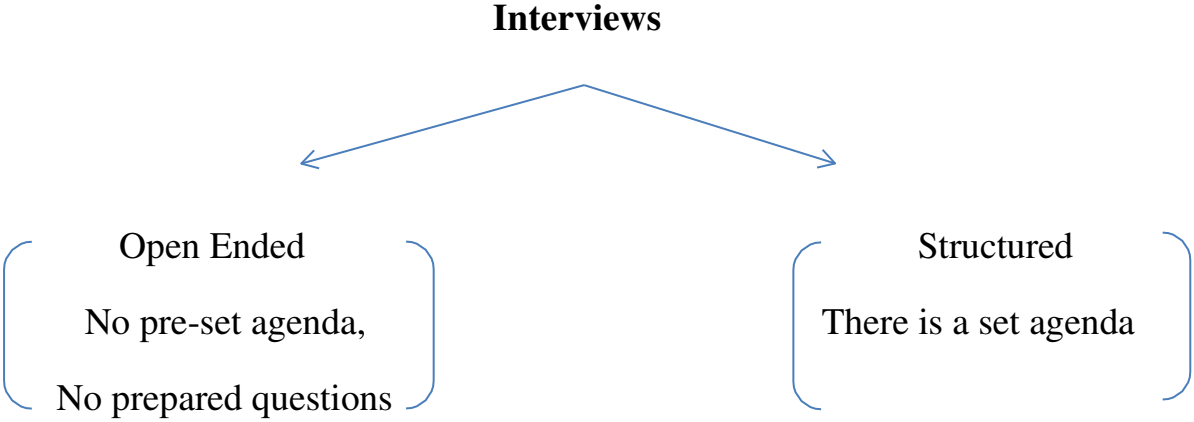
Benefits of Feasibility Study

1. A feasibility study gets you a clear-cut idea of whether the software will be a success even before you invest in the software.
2. It increases the focus of the software development team.
3. The software developer team may come across new opportunities.
4. Confirms the developer team whether to accept the software project or not.
5. Helps in evaluating the success rate of the software.

Requirement elicitation

- This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available.
- Requirement Elicitation is an activity that helps us to understand what problem has to be solved and what customers expect from the software.

- **Requirement Elicitation Method:**



What are the type of people in the interview?

1. Entry level Personnel: They do not have much domain knowledge but they have new ideas/perspective.

2. Mid-level Stakeholders: Experienced people with good domain knowledge and are interested in our project.

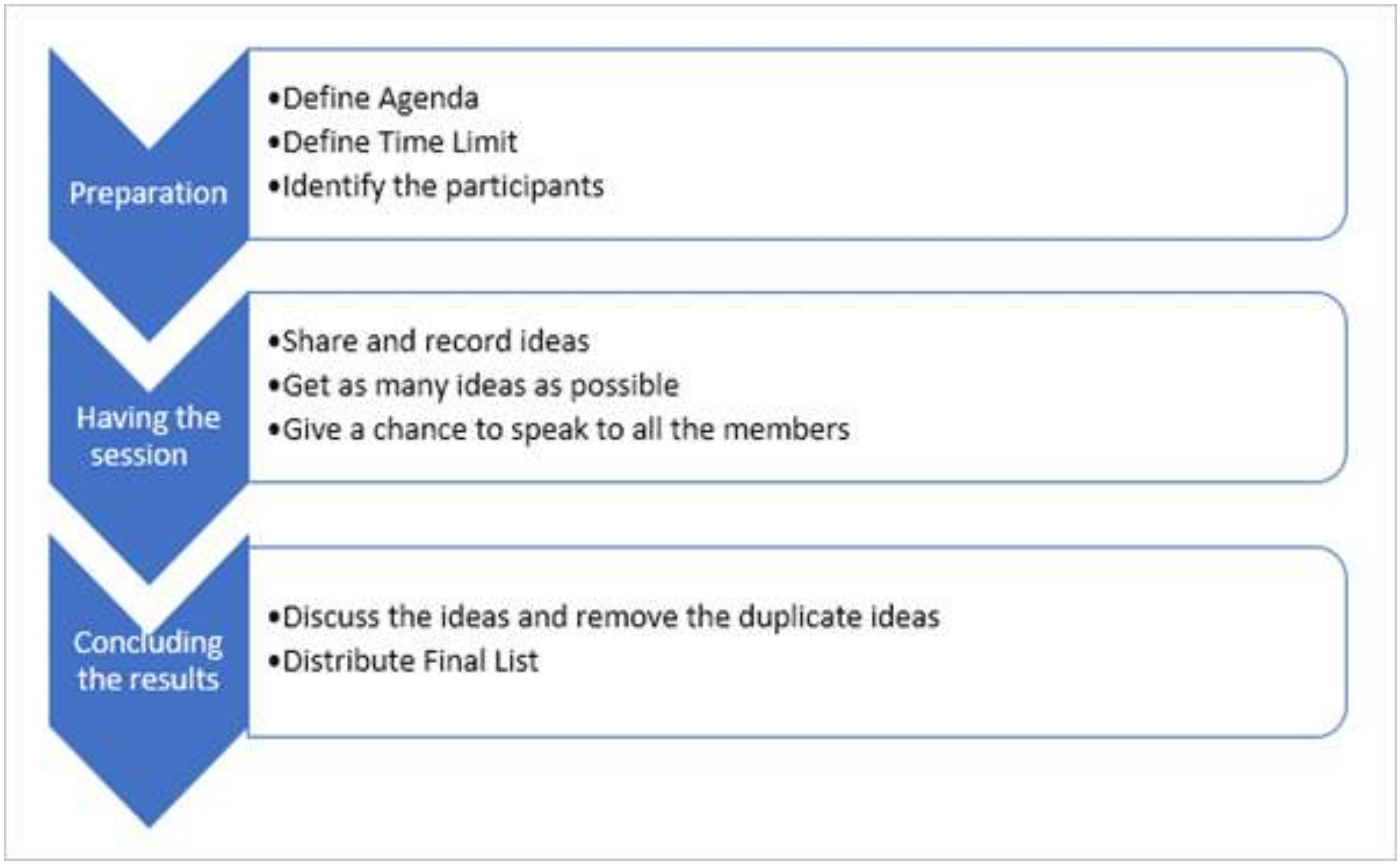
3. Managers and Higher Management: Highly experienced and give useful insight about the project.

4. Users and Developers of the software: The one who use and develop the Software.

Brainstorming Sessions:

- It is a group technique.
- It is intended to generate lots of new ideas hence providing a platform to share views.
- A highly trained facilitator is required to handle group bias and group conflicts.
- Every idea is documented so that everyone can see it.
- Finally, a document is prepared which consists of the list of requirements and their priority if possible.

Brainstorming can be described in the following phases:



Document Analysis/Review:

- Document analysis includes reviewing the business plans, technical documents, problem reports, existing requirement documents, etc. This is useful when the plan is to update an existing system. This technique is useful for migration projects.
- This analysis also helps when the person who has prepared the existing documentation is no longer present in the system.

Prototyping:

One of the most important phases of the requirements elicitation process, prototyping enables business owners and end-users to visualize realistic models of applications before they are finally developed. Prototyping helps generate early feedback, and it boosts stakeholder participation in requirements elicitation.

Survey/Questionnaires:

When multiple Subject Matter Experts and stakeholders are involved in a project, business analysts conduct a survey for the elicitation of requirements. Everyone involved is given a questionnaire to fill out.

Subsequently, the responses are analyzed to refine the requirements. Surveys are less expensive than other requirements elicitation techniques, easy to administer, and can produce both qualitative and quantitative results.

Software Requirement Specification

- Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources.

- A complete Software Requirement Specifications should be:

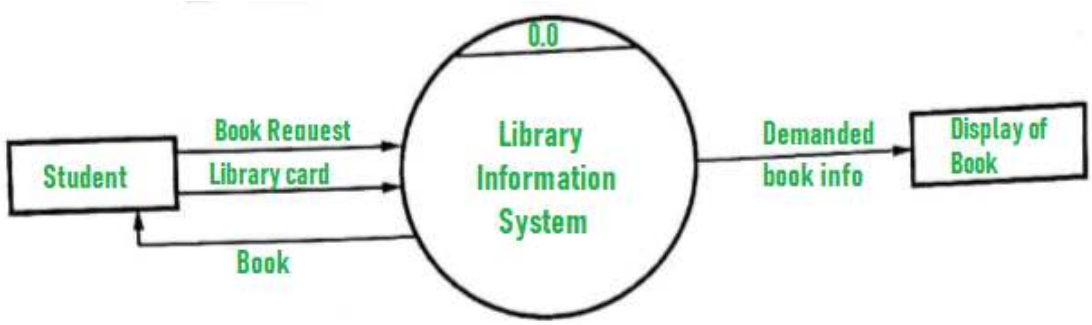
1. Clear
2. Correct
3. Consistent
4. Coherent
5. Comprehensible
6. Modifiable
7. Verifiable
8. Prioritized
9. Unambiguous
10. Traceable
11. Credible source

- The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

• **Data Flow Diagrams:** Data Flow Diagrams (DFDs) are used widely for modeling the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a data flow graph or bubble chart.

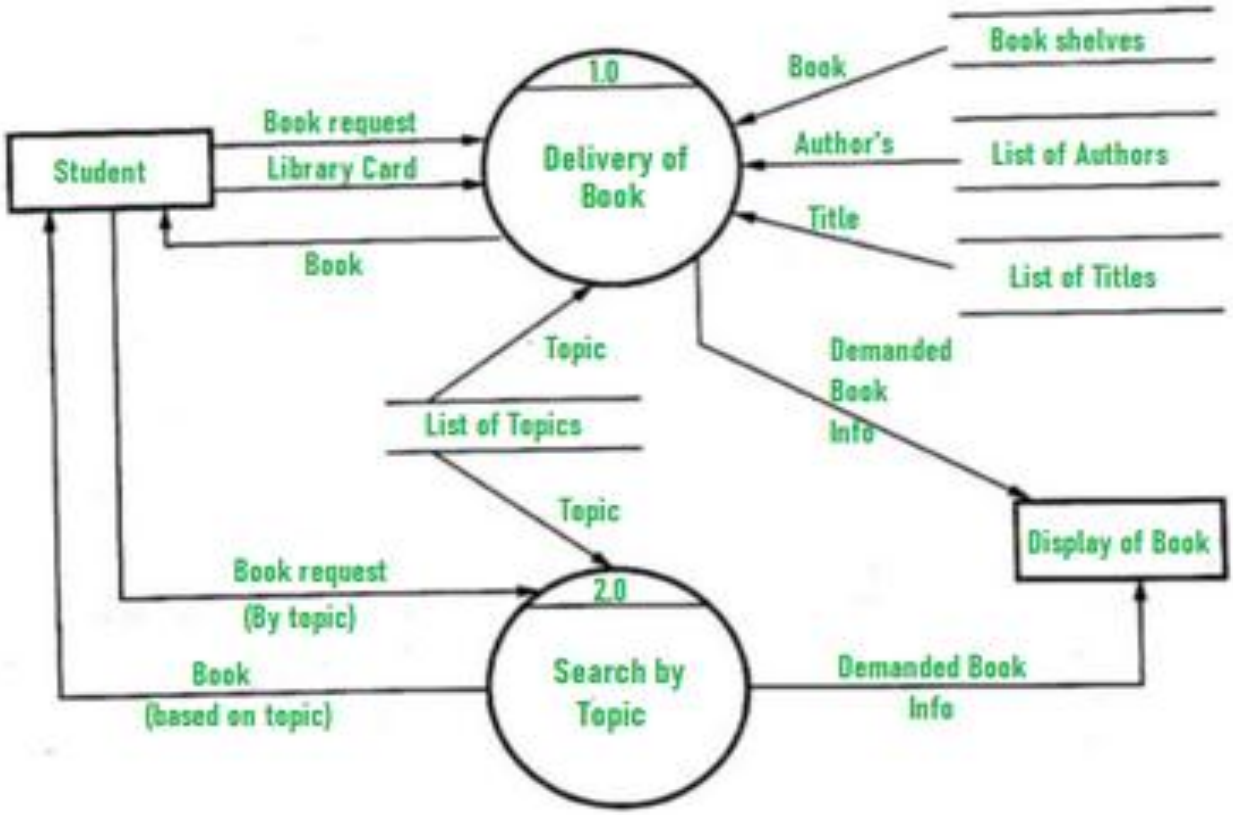
0-level DFD

• It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.



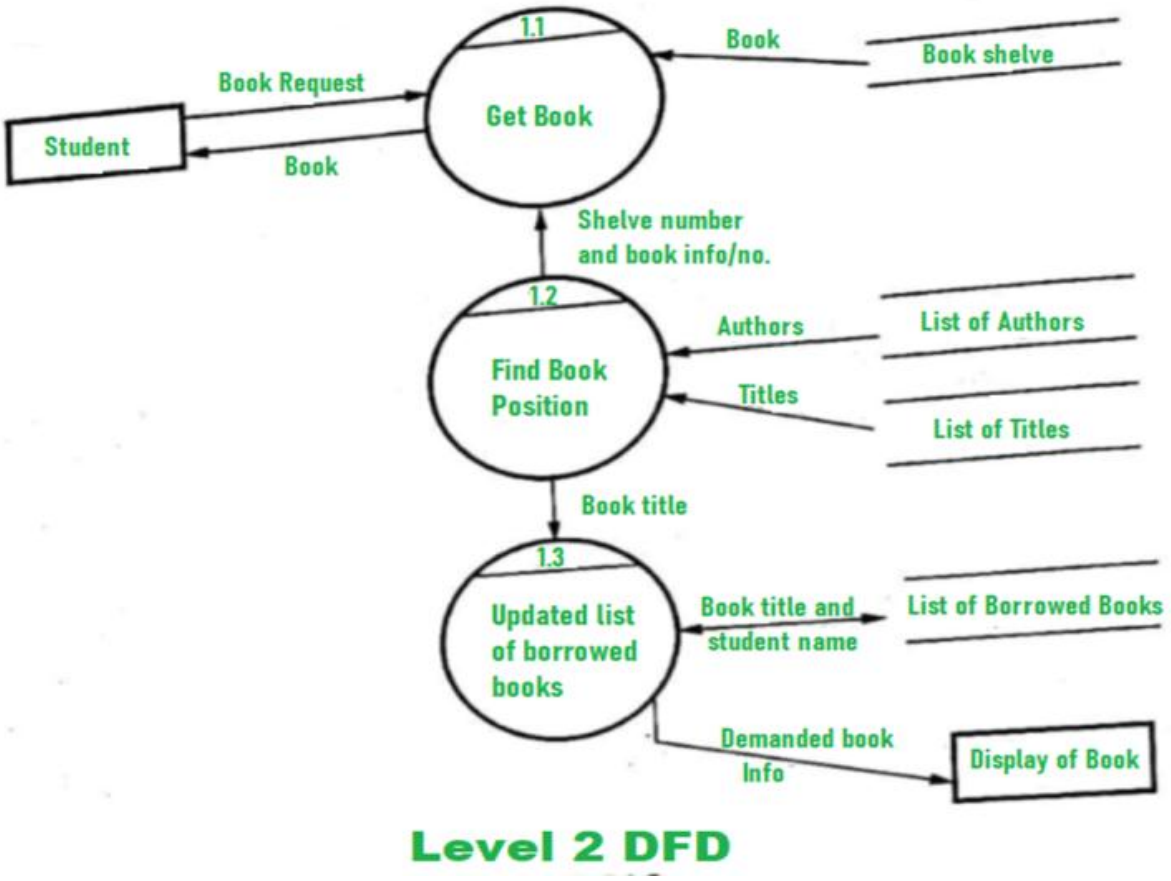
1-level DFD

• In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into sub processes.



2-level DFD

• 2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system's functioning



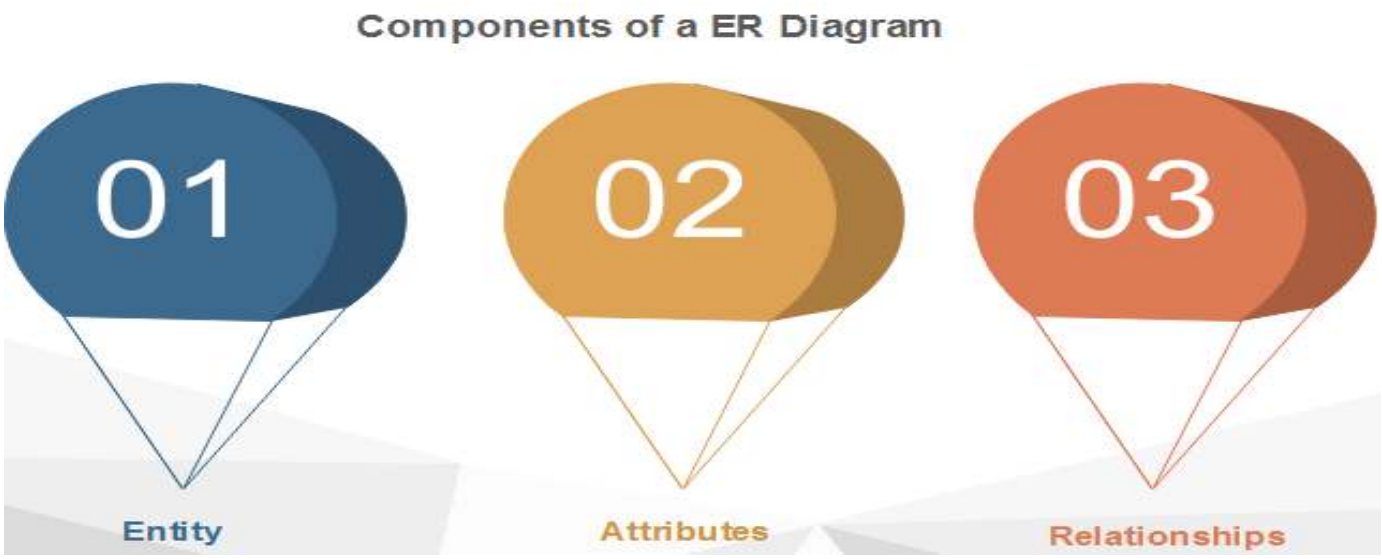
• **Data Dictionaries:** Data Dictionaries are simply repositories to store information about all data items defined in DFDs.

• **Computer-aided software engineering** (CASE) tools is used to maintain data dictionary as it captures the data items appearing in a DFD automatically to generate the data dictionary.

• In Software Engineering, the data dictionary contains the following information:

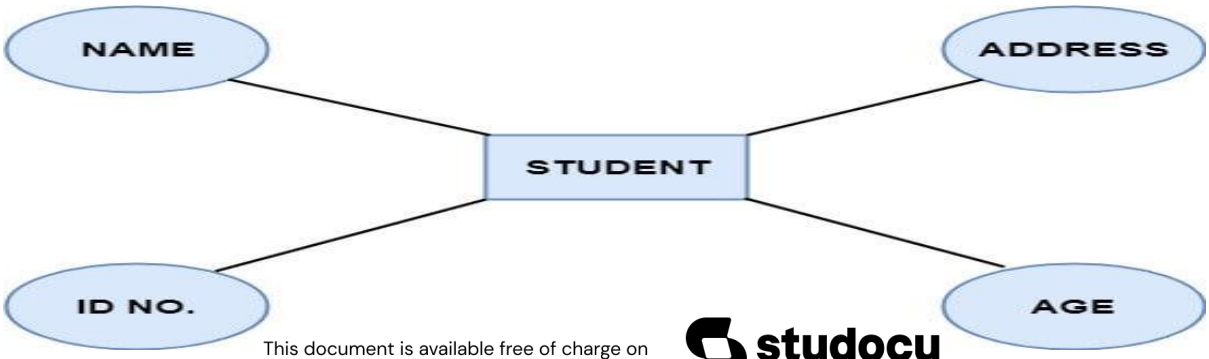
1. **Name of the item:** It can be your choice.
2. **Aliases:** It represents another name.
3. **Description:** Description of what the actual text is all about.
4. **Related data items:** with other data items.
5. **Range of values:** It will represent all possible answers.

• **Entity-Relationship Diagrams:** Another tool for requirement specification is the entity-relationship diagram, often called an "***E-R diagram***." It is a detailed logical representation of the data forthe organization.



• **Entity:** An entity can be a real-world object that can be merely identifiable. An entity is denoted as a rectangle in an ER diagram. For example, in a school database, students, teachers, classes, and courses offered can be treated as entities. All these entities have some attributes or properties that give them their identity.

• **Attributes:** Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.



- **There are four types of Attributes:**

1. Key attribute
2. Composite attribute
3. Single-valued attribute
4. Multi-valued attribute
5. Derived attribute

- **Relationships:** The association among entities is known as relationship.

Software Requirement Validation

- After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs.
- Requirements can be the check against the following conditions –
- If they can practically implement
- If they are correct and as per the functionality and specially of software
- If there are any ambiguities
- If they are full
- If they can describe
- **Requirements reviews**: The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.
- **Prototyping :** This involves developing an executable model of a system and using this with end-users and customers to see if it meets their needs and expectations. Stakeholders experiment with the system and feedback requirements changes to the development team.

• **Test-case generation**: Requirements should be testable. If the tests for the requirements are devised as part of the validation process, this often reveals requirements problems. If a test is difficult or impossible to design, this usually means that the requirements will be difficult to implement and should be reconsidered. Developing tests from the user requirements before any code is written is an integral part of test-driven development.

Software Requirement Management

• Requirement management is the process of managing changing requirements during the requirements engineering process and system development.

• New requirements emerge during the process as business needs a change, and a better understanding of the system is developed.

• Various activities performed during user needs review processes are:

1. Plan a review
2. Review meeting
3. Follow-up actions
4. Checking for redundancy
5. Completeness
6. Consistency

INFORMATION MODELLING

• It is used by software engineers and website designers to build an effective platform that is easy to use and navigate.

• If engineer or designer fails to build an information model or creates a poor one then many users will find website or program lacks intuitive features and navigation may be sloppy causing users to become frustrated.

• Most of the models have main domain at top and deeper domain at bottom.

• Engineer must plan for what user wants out of a program or website to make it effective.

Format of SRS

• SRS is developed which describes requirements of software that may include changes and modifications that is needed to be done to increase quality of product and to satisfy customer's demand.

1) Introduction :– (i) Purpose of this Document – At first, main aim of why this document is necessary and what's purpose of document is explained and described.

(ii) Scope of this document – In this, overall working and main objective of document and what value it will provide to customer is described and explained. It also includes a description of development cost and time required.

(iii) Overview – In this, description of product is explained. It's simply summary or overall review of product.

2)General description : In this, general functions of product which includes objective of user, a user characteristic, features, benefits, about why its importance is mentioned. It also describes features of user community.

3Functional Requirements : In this, possible outcome of software system which includes effects due to operation of program is fully explained. All functional requirements which may include calculations, data processing, etc. are placed in a ranked order.

4)Interface Requirements : In this, software interfaces which mean how software program communicates with each other or users either in form of any language, code, or message are fully described and explained. Examples can be shared memory, data streams, etc.

5)Performance Requirements : In this, how a software system performs desired functions under specific condition is explained. It also explains required time, required memory, maximum error rate,etc.

6)Design Constraints : In this, constraints which simply means limitation or restriction are specified and explained for design team. Examples may include use of a particular algorithm, hardware and software limitations, etc.

7)Non-Functional Attributes : In this, non-functional attributes are explained that are required by software system for better performance. An example may include Security, Portability, Reliability, Reusability, Application compatibility, Data integrity, Scalability capacity, etc.

8)Preliminary Schedule and Budget : In this, initial version and budget of project plan are explained which include overall time duration required and overall cost required for development of project.

9)Appendices : In this, additional information like references from where information is gathered, definitions of some specific terms, acronyms, abbreviations, etc. are given and explained.

Uses of SRS document

- Development team requires it for developing product according to the need.
- Test plans are generated by testing group based on the describe external behavior.
- Maintenance and support staff need it to understand what the software product is supposed to do.
- Project manager base their plans and estimates of schedule, effort and resources on it.
- Customer rely on it to know that product they can expect.
- As a contract between developer and customer.
- For documentation purpose.

What is Software Quality Assurance?

- **Software Quality Assurance (SQA)** is a process that assures that all software engineering processes, methods, activities, and work items are monitored and comply with the defined standards. These defined standards could be one or a combination of anything like ISO 9000, CMMI model, ISO15504, etc.
- SQA incorporates all software development processes starting from defining requirements to coding until release. Its prime goal is to ensure quality.

Software Quality Assurance Plan :-

It comprises the procedures, techniques, and tools that are employed to make sure that a product or service aligns with the requirements defined in the SRS (Software Requirement Specification).

Software Quality Assurance has:

1) A quality management approach
2) Formal technical reviews
3) Multi testing strategy
4) Effective software engineering technology
5) Measurement and reporting mechanism

Major Software Quality Assurance Activities:

A.SQA Management Plan:

Make a plan for how you will carry out the SQA throughout the project. Think about which set of software engineering activities are the best for project. Check level of SQA team skills.

B. Set The Check Points:

SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.

C. Multi testing Strategy:

Do not depend on a single testing approach. When you have a lot of testing approaches available use them.

D. Measure Change Impact:

The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change, check the compatibility of this fix with whole project.

E. Manage Good Relations:

In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of sqa team with programmers team will impact directly and badly on project. Don't play politics.

Disadvantage of SQA:-

There are a number of disadvantages of quality assurance. Some of them include adding more resources, employing more workers to help maintain quality and so much more.

Benefits of Software Quality Assurance (SQA):

1. SQA produces high quality software.
2. High quality application saves time and cost. qSQA is beneficial for better reliability.
3. SQA is beneficial in the condition of no maintenance for a long time.
4. High quality commercial software increase market share of company.
5. Improving the process of creating software. qImproves the quality of the software.

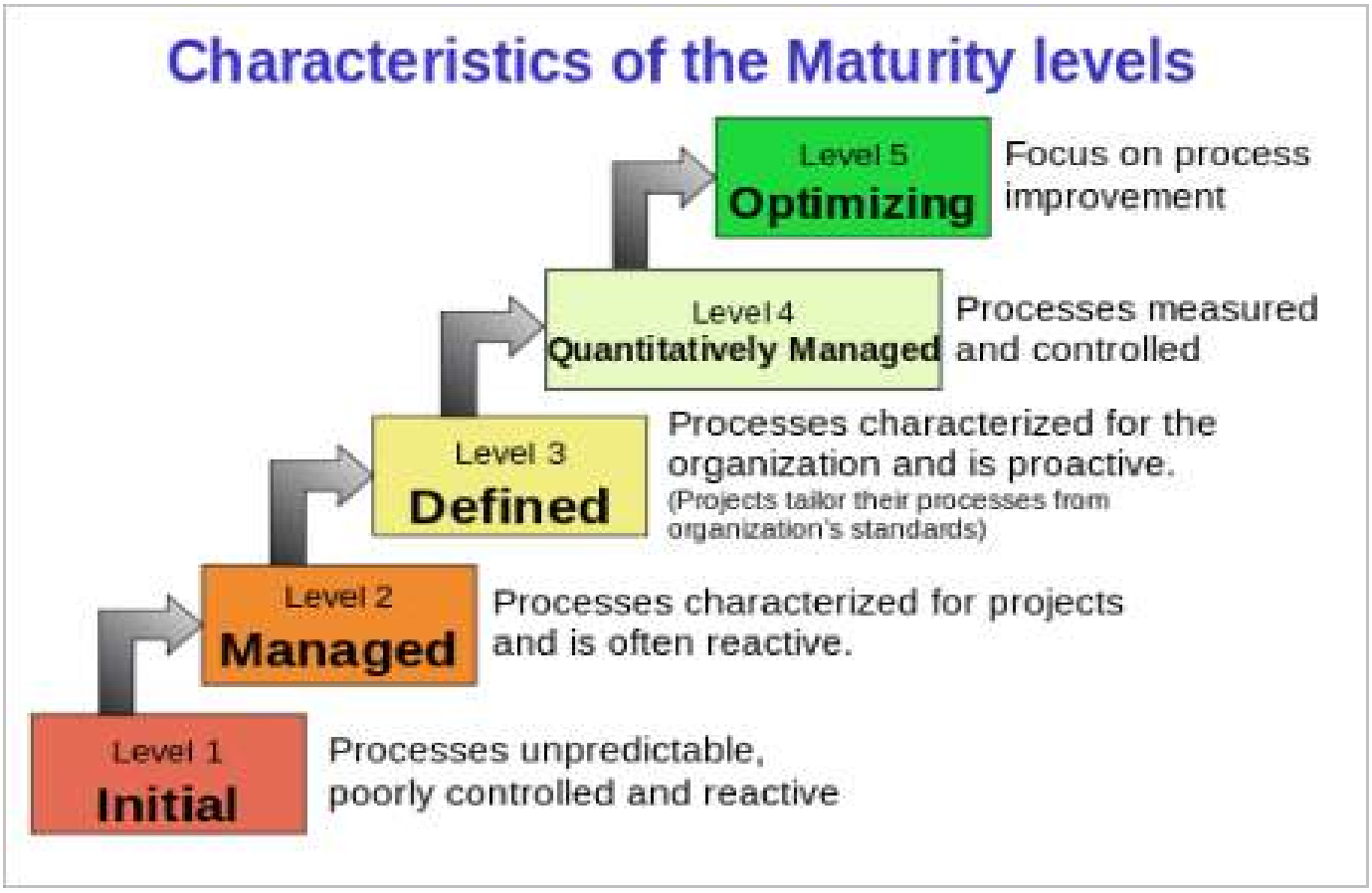
Software Quality Assurance Standards

1) ISO 9000 :- Based on seven quality management principles that help organizations ensure that their products or services are aligned with customer needs.

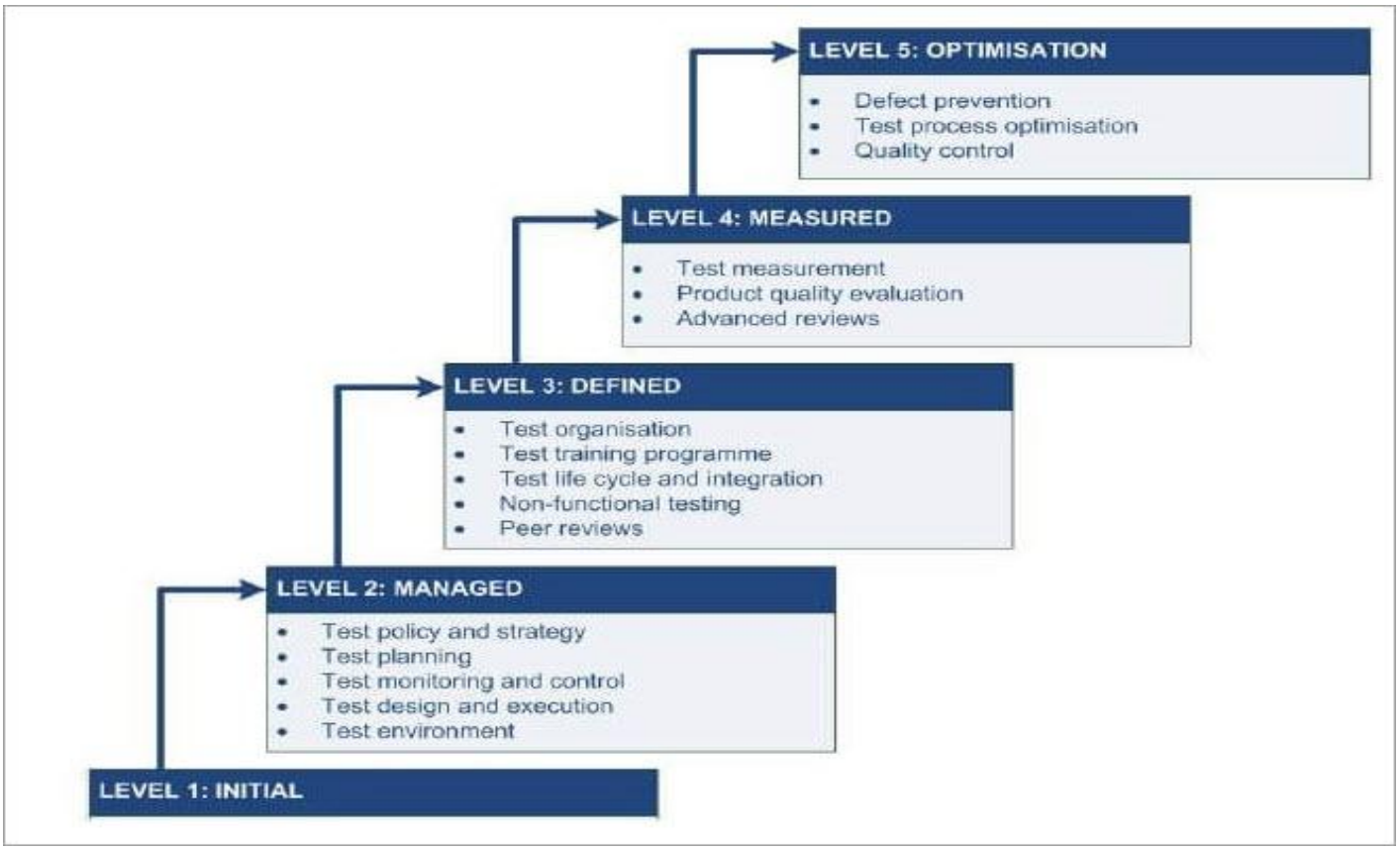
7 principles of ISO 9000 are depicted in the below image:

2) CMMI level: CMMI stands for **Capability Maturity Model Integration**. This model originated in software engineering. It can be employed to direct process improvement throughout a project, department, or entire organization.

5 CMMI levels and their characteristics are described in the below image:



3) Test Maturity Model integration (TMMi): Based on CMMi, this model focuses on maturity levels in software quality management and testing.



SQA Techniques

SQA Techniques include:

1) Auditing: Auditing is the inspection of the work products and its related information to determine if a set of standard processes were followed or not.

2) Reviewing: A meeting in which the software product is examined by both internal and external stakeholders to seek their comments and approval.

3) Code Inspection: It is the most formal kind of review that does static testing to find bugs and avoid defect seepage into the later stages. It is done by a trained mediator/peer and is based on rules, checklists, entry and exit criteria. The reviewer should not be the author of the code.

4) Design Inspection: Design inspection is done using a checklist that inspects the below areas of software design:

– General requirements and design

– Functional and Interface specifications

– Conventions

– Requirement traceability

– Structures and interfaces

– Logic

– Performance

– Error handling and recovery

– Testability, extensibility

– Coupling and cohesion

5) Simulation: A simulation is a tool that models a real-life situation in order to virtually examine the behavior of the system under study. In cases when the real system cannot be tested directly, simulators are great sandbox system alternatives.

6) Functional Testing: It is a QA technique that validates what the system does without considering how it does it. Black Box testing mainly focuses on testing the system specifications or features.

7) Standardization: Standardization plays a crucial role in quality assurance. This decreases ambiguity and guesswork, thus ensuring quality.

8) Static Analysis: It is a software analysis that is done by an automated tool without executing the program. Software metrics and reverse engineering are some popular forms of static analysis. In newer teams, static code analysis tools such as Sonar Cube, Vera Code, etc. are used.

9)Walkthroughs: A software walkthrough or code walkthrough is a peer review where the developer guides the members of the development team to go through the product, raise queries, suggest alternatives, and make comments regarding possible errors, standard violations, or any other issues.

10) Unit Testing: This is a White Box Testing techniquewhere complete code coverage is ensured by executing each independent path, branch, and condition at least once.

11) Stress Testing: This type of testing is done to check how robust a system is by testing it under heavy load i.e. beyond normal conditions.