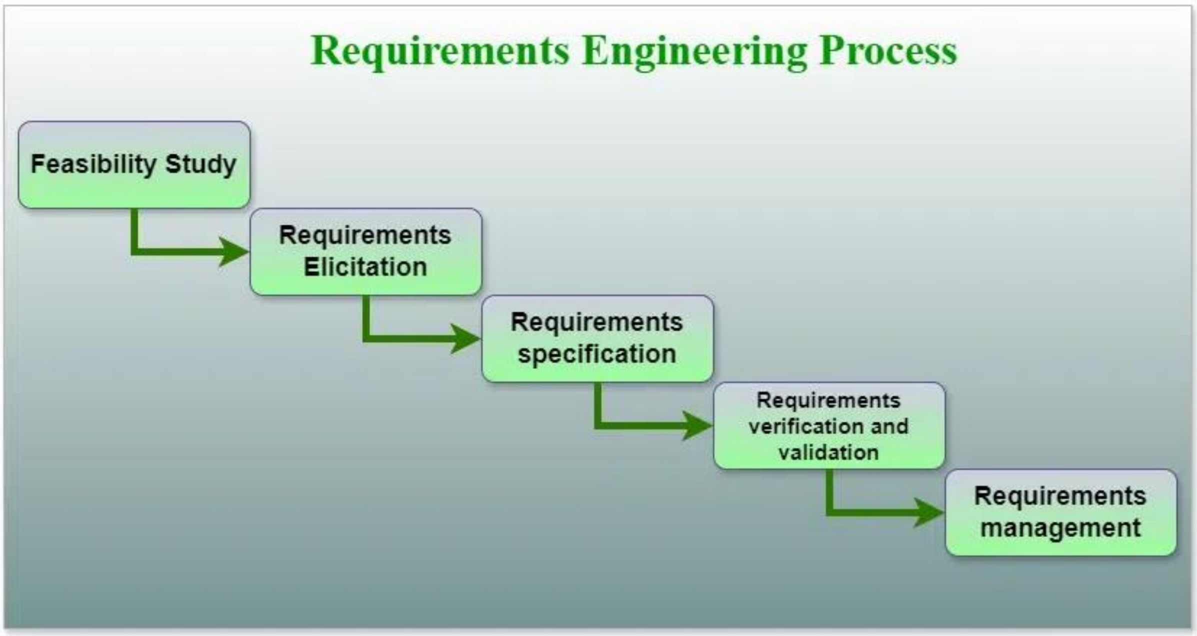
Software Engineering

UNIT 2

1. Requirements Engineering

Requirements Engineering is the process of identifying, eliciting, analyzing, specifying, validating, and managing the needs and expectations of stakeholders for a software system. In this article, we'll learn about its process, advantages, and disadvantages. A systematic and strict approach to the definition, creation, and verification of requirements for a software system is known as requirements engineering. To guarantee the effective creation of a software product, the requirements engineering process entails several tasks that help in understanding, recording, and managing the demands of stakeholders.


Requirements Engineering Process
Feasibility Study
Requirements Elicitation
Requirements specification
Requirements verification and validation
Requirements management

1.1. Feasibility Study

The feasibility study mainly concentrates on below five mentioned areas below. Among these Economic Feasibility Study is the most important part of the feasibility analysis and the Legal Feasibility Study is less considered feasibility analysis.

- Technical Feasibility: In Technical Feasibility current resources both hardware software along required technology are analyzed/assessed to develop the project. This technical feasibility study reports whether there are correct required resources and technologies that will be used for project development. Along with this, the feasibility study also analyzes the technical skills and capabilities of the technical team, whether existing technology can be used or not, whether maintenance and up-gradation are easy or not for the chosen technology, etc.
- Operational Feasibility: In Operational Feasibility degree of providing service to requirements is analyzed along with how easy the product will be to operate and maintain after deployment. Along with this other operational scopes are determining the usability of the product, Determining suggested solution by the software development team is acceptable or not, etc.

- Economic Feasibility: In the Economic Feasibility study cost and benefit of the project are analyzed. This means under this feasibility study a detailed analysis is carried out will be cost of the project for development which includes all required costs for final development hardware and software resources required, design and development costs operational costs, and so on. After that, it is analyzed whether the project will be beneficial in terms of finance for the organization or not.

1.2. Requirements Elicitation

It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of the same type, standards, and other stakeholders of the project. The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping, etc. Some of these are discussed here. Elicitation does not produce formal models of the requirements understood. Instead, it widens the domain knowledge of the analyst and thus helps in providing input to the next stage. Requirements elicitation is the process of gathering information about the needs and expectations of stakeholders for a software system. This is the first step in the requirements engineering process and it is critical to the success of the software development project. The goal of this step is to understand the problem that the software system is intended to solve and the needs and expectations of the stakeholders who will use the system. Several techniques can be used to elicit requirements, including:

- Interviews: These are one-on-one conversations with stakeholders to gather information about their needs and expectations.
- Surveys: These are questionnaires that are distributed to stakeholders to gather information about their needs and expectations.
- Focus Groups: These are small groups of stakeholders who are brought together to discuss their needs and expectations for the software system.
- Observation: This technique involves observing the stakeholders in their work environment to gather information about their needs and expectations.
- Prototyping: This technique involves creating a working model of the software system, which can be used to gather feedback from stakeholders and to validate requirements.

It’s important to document, organize, and prioritize the requirements obtained from all these techniques to ensure that they are complete, consistent, and accurate.

1.3. Requirements Specification

This activity is used to produce formal software requirement models. All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality. During specification, more knowledge about the problem may be required which can again trigger the elicitation process. The models used at this stage include ER diagrams, data flow diagrams(DFDs), function decomposition diagrams(FDDs), data dictionaries, etc.

Requirements specification is the process of documenting the requirements identified in the analysis step in a clear, consistent, and unambiguous manner. This step also involves prioritizing and grouping the requirements into manageable chunks. The goal of this step is to create a clear and comprehensive document that describes the requirements for the software system. This document should be understandable by both the development team and the stakeholders. Several types of requirements are commonly specified in this step, including

- Functional Requirements: These describe what the software system should do. They specify the functionality that the system must provide, such as input validation, data storage, and user interface.
- Non-Functional Requirements: These describe how well the software system should do it. They specify the quality attributes of the system, such as performance, reliability, usability, and security.
- Constraints: These describe any limitations or restrictions that must be considered when developing the software system.
- Acceptance Criteria: These describe the conditions that must be met for the software system to be considered complete and ready for release.

To make the requirements specification clear, the requirements should be written in a natural language and use simple terms, avoiding technical jargon, and using a consistent format throughout the document. It is also important to use diagrams, models, and other visual aids to help communicate the requirements effectively. Once the requirements are specified, they must be reviewed and validated by the stakeholders and development team to ensure that they are complete, consistent, and accurate.

1.4. Requirements Verification and Validation

Requirements verification and validation (V&V) is the process of checking that the requirements for a software system are complete, consistent, and accurate and that they meet the needs and expectations of the stakeholders. The goal of V&V is to ensure that the software system being developed meets the requirements and that it is developed on time, within budget, and to the required quality.

- Verification: It refers to the set of tasks that ensures that the software correctly implements a specific function. Verification is checking that the requirements are complete, consistent, and accurate. It involves reviewing the requirements to ensure that they are clear, testable, and free of errors and inconsistencies. This can include reviewing the requirements document, models, and diagrams, and holding meetings and walkthroughs with stakeholders.
- Validation: It refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements. If requirements are not validated, errors in the requirement definitions would propagate to the successive stages resulting in a lot of modification and rework. Validation is the process of checking that the requirements meet the needs and expectations of the stakeholders. It involves testing the requirements to ensure that they are valid and that the software system being developed will meet the needs of the stakeholders. This can include testing the software system through simulation, testing with prototypes, and testing with the final version of the software.

The requirements should be consistent with all the other requirements i.e. no two requirements should conflict with each other.

- The requirements should be complete in every sense.
- The requirements should be practically achievable.
- Reviews, buddy checks, making test cases, etc. are some of the methods used for this.

Verification and Validation is an iterative process that occurs throughout the software development life cycle. It is important to involve stakeholders and the development team in the V&V process to ensure that the requirements are thoroughly reviewed and tested. It's important to note that V&V is not a one-time process, but it should be integrated and continue throughout the software development process and even in the maintenance stage.

Verification	**Validation**
It includes checking documents, design, coc programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.
It does *not* include the execution of the code.	It includes the execution of the code.
Methods used in verification are r walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing Box Testing and non-functional testing.
It checks whether the software confor specifications or not.	It checks whether the software meets the requireme expectations of a customer or not.
It can find the bugs in the early stage development.	It can only find the bugs that could not be found verification process.
The goal of verification is application and s architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files performed by human.	It consists of execution of program and is perfor computer.
Verification refers to the set of activities that software correctly implements the specific func	Validation refers to the set of activities that ensure software that has been built is traceable to cu requirements.
After a valid and complete specification the veri starts.	Validation begins as soon as project starts.
Verification is for prevention of errors.	Validation is for detection of errors.
Verification is also termed as white box testing testing as work product goes through reviews.	Validation can be termed as black box testing or d testing as work product is executed.
Verification finds about 50 to 60% of the defec	Validation finds about 20 to 30% of the defects.
Verification is based on the opinion of reviev may change from person to person.	Validation is based on the fact and is often stable.
Verification is about process, standard and guid	Validation is about the product.

1.5. Requirements Management

Requirement management is the process of analyzing, documenting, tracking, prioritizing, and agreeing on the requirement and controlling the communication with relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible to incorporate changes in requirements specified by the end users at later stages too. Modifying the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process. Requirements management is the process of managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant. The goal of requirements management is to ensure that the software system being developed meets the needs and expectations of the stakeholders and that it is developed on time, within budget, and to the required quality. Several key activities are involved in requirements management, including:

- Tracking and controlling changes: This involves monitoring and controlling changes to the requirements throughout the development process, including identifying the source of the change, assessing the impact of the change, and approving or rejecting the change.

- Version control: This involves keeping track of different versions of the requirements document and other related artifacts.
- Traceability: This involves linking the requirements to other elements of the development process, such as design, testing, and validation.
- Communication: This involves ensuring that the requirements are communicated effectively to all stakeholders and that any changes or issues are addressed promptly.
- Monitoring and reporting: This involves monitoring the progress of the development process and reporting on the status of the requirements.

Requirements management is a critical step in the software development life cycle as it helps to ensure that the software system being developed meets the needs and expectations of stakeholders and that it is developed on time, within budget, and to the required quality. It also helps to prevent scope creep and to ensure that the requirements are aligned with the project goals. Tools Involved in Requirement Engineering

- Observation report
- Questionnaire (survey, poll)
- Use cases
- User stories
- Requirement workshop
- Mind mapping
- Roleplaying
- Prototyping

Advantages of Requirements Engineering Process

- Helps ensure that the software being developed meets the needs and expectations of the stakeholders
- Can help identify potential issues or problems early in the development process, allowing for adjustments to be made before significant
- Helps ensure that the software is developed in a cost-effective and efficient manner
- Can improve communication and collaboration between the development team and stakeholders
- Helps to ensure that the software system meets the needs of all stakeholders.
- Provides an unambiguous description of the requirements, which helps to reduce misunderstandings and errors.
- Helps to identify potential conflicts and contradictions in the requirements, which can be resolved before the software development process begins.
- Helps to ensure that the software system is delivered on time, within budget, and to the required quality standards.
- Provides a solid foundation for the development process, which helps to reduce the risk of failure.

Disadvantages of Requirements Engineering Process

- Can be time-consuming and costly, particularly if the requirements-gathering process is not well-managed
- Can be difficult to ensure that all stakeholders' needs and expectations are taken into account
- It Can be challenging to ensure that the requirements are clear, consistent, and complete
- Changes in requirements can lead to delays and increased costs in the development process.

- As a best practice, Requirements engineering should be flexible, adaptable, and should be aligned with the overall project goals.
- It can be time-consuming and expensive, especially if the requirements are complex.
- It can be difficult to elicit requirements from stakeholders who have different needs and priorities.
- Requirements may change over time, which can result in delays and additional costs.
- There may be conflicts between stakeholders, which can be difficult to resolve.
- It may be challenging to ensure that all stakeholders understand and agree on the requirements.

Stages in Software Engineering Process

Requirements engineering is a critical process in software engineering that involves identifying, analyzing, documenting, and managing the requirements of a software system. The requirements engineering process consists of the following stages:

- Elicitation: In this stage, the requirements are gathered from various stakeholders such as customers, users, and domain experts. The aim is to identify the features and functionalities that the software system should provide.
- Analysis: In this stage, the requirements are analyzed to determine their feasibility, consistency, and completeness. The aim is to identify any conflicts or contradictions in the requirements and resolve them.
- Specification: In this stage, the requirements are documented in a clear, concise, and unambiguous manner. The aim is to provide a detailed description of the requirements that can be understood by all stakeholders.
- Validation: In this stage, the requirements are reviewed and validated to ensure that they meet the needs of all stakeholders. The aim is to ensure that the requirements are accurate, complete, and consistent.
- Management: In this stage, the requirements are managed throughout the software development lifecycle. The aim is to ensure that any changes or updates to the requirements are properly documented and communicated to all stakeholders.

Effective requirements engineering is crucial to the success of software development projects. It helps ensure that the software system meets the needs of all stakeholders and is delivered on time, within budget, and to the required quality standards.

2. Requirement reviews and management

Requirements are mandatory in terms of software development. Requirements are the basics that take forward the development procedure. Requirements enhance the existing project to a different level. The software cannot progress without the participation of user input. User input like feedback and product review comes as the major requirements in the development work. In short, Requirement review is the practice of scanning the software errors to make the industry user-friendly for all.

Software in the current time is so Advanced that the act of requirement review holds greater importance in software development. The pursuance of requirement reviews helps to have a clear peek into the space of the software industry. The requirement reviews call attention to the only chance of finding quality reviews. So keeping in mind that there are problems and solutions to everything, a requirement review needs a good performance. Importance of performing requirement review:

- The performance of requirement review helps to radiate the precise and correct data to the consumers and users.
- It helps to have a quick tour of the Existing project to see whether or not it is going in the right direction.
- It helps to provide practical instructions and helps make decisions accordingly.

Methods of performing requirement review:

- Team consultation: Suggestion matters also matter the way of performance. Teamwork goes hand in hand. When there are people to offer suggestions, give appropriate guidelines, and supervise in a team. There is no doubt about the project getting mismanaged. Reaching out to the team/individual who has better insights into requirement review works the best way.

- Understanding the user's requirement: Recognize the user's needs and go all out in understanding them. Requirements keep on changing with time. So, When you have collected a list of things that a user requires in the current time. There you found a way to go about it. To get exact information on their requirements, Asking for feedback is Important.

- Finding measures to software problem: The occurrence of software problems is predictable. Errors and defects are bound to take place in software development. In this context, Rather than making a fuss about the Problems, developers should find solutions to satisfy the requirements. The requirement review not only meets the expectations of users but also the standard of the entire industry.

Advantages of performing requirement reviews:

- Requirement reviews accord the developers a motive and structure to carry out the project further.
- Group collaboration is the highlight. Group work saves time.
- Therefore, the developers can utilize the saved time in rechecking and reconfirming the processing work to take it ahead.

Disadvantages of performing requirement reviews:

- Lack of attention acts as a hindrance. When a team does not listen to each other in a meeting room because of disagreement on matters, it emerges as a sign of unprofessional and uncoordinated work.
- At times, the Review cannot be accurate. So, If you fail in assembling the precise information, it can be an obstacle for the developers and the industry.

3. Documentation

Software documentation is a written piece of text that is often accompanied by a software program. This makes the life of all the members associated with the project easier. It may contain anything from API documentation, build notes or just help content. It is a very critical process in software development. It's primarily an integral part of any computer code development method. Moreover, computer code practitioners are a unit typically concerned with the worth, degree of usage, and quality of the actual documentation throughout the development and its maintenance throughout the total method. Motivated by the requirements of Novatel opposition, a world-leading company developing package in support of worldwide navigation satellite system, and based mostly on the results of a former systematic mapping studies area unit aimed at a higher understanding of the usage and therefore the quality of varied technical documents throughout computer code development and their maintenance. For example, before the development of any software product requirements are documented which is called Software Requirement Specification (SRS). Requirement gathering is considered a stage of Software Development Life Cycle (SDLC). Another example can be a user manual that a user refers to for installing, using, and providing maintenance to the software application/product.

Types:

- Requirement Documentation: It is the description of how the software shall perform and which environment setup would be appropriate to have the best out of it. These are generated while the software is under development and is supplied to the tester groups too.
- Architectural Documentation: Architecture documentation is a special type of documentation that concerns the design. It contains very little code and is more focused on the components of the system, their roles, and working. It also shows the data flow throughout the system.
- Technical Documentation: These contain the technical aspects of the software like API, algorithms, etc. It is prepared mostly for software devs.
- End-user Documentation: As the name suggests these are made for the end user. It contains support resources for the end user.

Purpose of Documentation

Due to the growing importance of computer code necessities, the method of crucial them needs to be effective to notice desired results. As to such determination of necessities is often beneath sure regulation and pointers that area unit core in getting a given goal. These all imply that computer code necessities area unit expected to alter thanks to the ever ever-changing technology within the world. However, the very fact that computer code information I'd obtained through development has to be modified within the wants of users and the transformation of the atmosphere area unit is inevitable. For a software engineer reliable documentation is typically a should the presence of documentation helps keep track of all aspects of associate applications, and it improves the standard of wares, it's the most focused area of unit development, maintenance, and information transfer to alternative developers. Productive documentation can build info simply accessible, offer a restricted range of user entry purposes, facilitate new users to learn quickly, alter the merchandise and facilitate chopping out the price.

Importance of software documentation: For a programmer reliable documentation is always a must the presence keeps track of all aspects of an application and helps in keeping the software updated.

Principles of Software Documentation: While writing or contributing into any software documentation, one must keep in mind the following set of 7-principles :

- Write from reader's point of view: It's important to keep in mind the targeted audience that will be learning, and working through the software's documentation to understand and implement the fully functional robust software application and even the ones who will be learning for the purpose of using the software. So, while writing a documentation it becomes very crucial to use the simplest language & domain related specific languages and terminologies. The structure of the documentation should be organized in a clearly viewable, navigable and understandable format.

- Avoid unnecessary repetition: While the idea of hyperlinking and backlinking may seem redundant at the moment, but it aids in avoiding the need of redundancy. The back-end database stores every piece of information as an individual unit and displays it in various different variety of context so redundancy at any point will not be maintainable and is considered a bad practice.

- Avoid ambiguity: Documentation contains a lot of information regarding the versatile functionalities of the software system, every part of it must be written with clear and precise knowledge while avoiding any conflicting information that might cause confusion to the reader. For example, if one terminology is used in different set of context than it must be explicitly defined what it means so to avoid any miscommunication. This aspect of the software documentation is very important to avoid any kind of conflicting knowledge between the stakeholders, developers and the maintainers.

- Follow a certain standard organization: In order to maintain the professionalism, accuracy, and precision of the document a certain set of principles must be followed taking reference from other software documentations that would aid in organizing and structuring the content of the documentation in a much productive and organized way.

- Record a Rationale: Rationale contains a comprehensive understanding of why a certain design or development decision was made. This part of our documentation is written & maintained by the developer or the designer itself for justification and verification for later needs. Rationale can be mentioned in the start or the end of the document although typically, it's in the start of the document.

- Keep the documentation updated but to an extent: This principle applies to the maintainers of the documentation of the software, because updates are made to the software on frequent intervals. The updates may contain some bug fixes, new feature addition or previous functionality maintenance. The maintainer of the documentation must only add the valuable content and avoid anything that doesn't fit and irrelevant for that particular time.

- Review documentation: The documentation consists of too many web-pages collectively holding a large chunk of information that's serving a sole purpose – educate and spread knowledge to anyone who is trying to understand or implement the software. While working with a lot of information it is important ta take feedback from senior architects and make any necessary changes aligning the documentation with its sole purpose depending on the type of documentation.

Advantages of software documentation

- The presence of documentation helps in keeping the track of all aspects of an application and also improves the quality of the software product.
- The main focus is based on the development, maintenance, and knowledge transfer to other developers.
- Helps development teams during development.
- Helps end-users in using the product.
- Improves overall quality of software product
- It cuts down duplicative work.
- Makes easier to understand code.
- Helps in establishing internal coordination in work.

Disadvantages of software documentation

- The documenting code is time-consuming.

- The software development process often takes place under time pressure, due to which many times the documentation updates don’t match the updated code.
- The documentation has no influence on the performance of an application.
- Documenting is not so fun, it’s sometimes boring to a certain extent.

4. Information modelling

An **information model** in software engineering is a representation of concepts and the relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse. Typically it specifies relations between kinds of things, but may also include relations with individual things. It can provide sharable, stable, and organized structure of information requirements or knowledge for the domain context.

5 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart. Standard symbols for DFDs are shown below:

- Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.
- Data Flow: A curved line shows the flow of data into or out of a process or data store.
- Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.
- Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

Levels in DFD

The DFD may be used to perform a system or software at any level of abstraction. Infact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

- 0-level DFDM- It is also known as fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood. It is essential to preserve the number of inputs and outputs between levels, this concept is called leveling by DeMacro.

- 1-level DFD- In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into sub processes.
- 2-Level DFD- 2-level DFD goes one process deeper into parts of 1-level DFD. It can be used to project or record the specific/necessary detail about the system's functioning.
- EXAMPLES: discussed in class ……for more information refer https://www.javatpoint.com/software-engineering-data-flow-diagrams

6 Entity-Relationship diagrams

ER-modeling is a data modeling method used in software engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling method are called Entity-Relationship Diagrams or ER diagrams or ERDs.

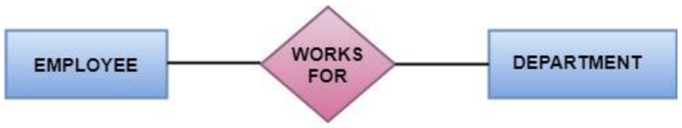
Purpose of ERD

- The database analyst gains a better understanding of the data to be contained in the database through the step of constructing the ERD.
- The ERD serves as a documentation tool.
- Finally, the ERD is used to connect the logical structure of the database to users. In particular, the ERD effectively communicates the logic of the database to users.

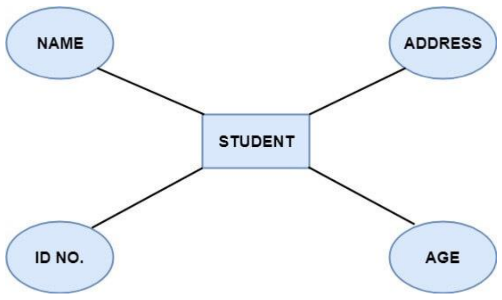
Components of an ER Diagrams

i) Entity- An entity can be a real-world object, either animate or inanimate, 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.

- Entity Set- An entity set is a collection of related types of entities. An entity set may include entities with attribute sharing similar values. For example, a Student set may contain all the students of a school; likewise, a Teacher set may include all the teachers of a school from all faculties. Entity set need not be disjoint.

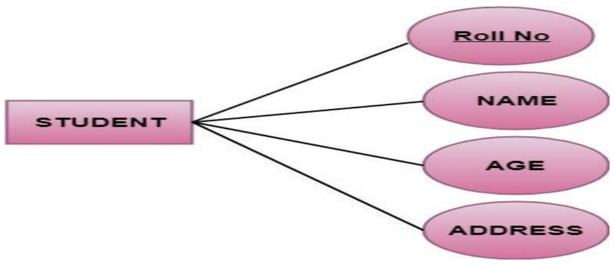


ii) Attributes- Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes. There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

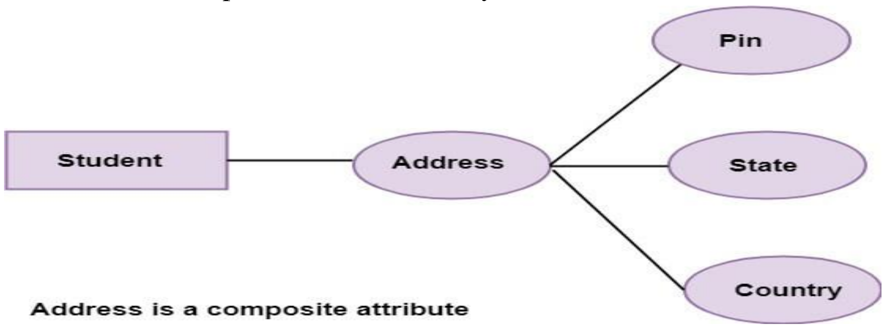


There are four types of Attributes:

a) Key attribute: Key is an attribute or collection of attributes that uniquely identifies an entity among the entity set. For example, the roll_number of a student makes him identifiable among students.

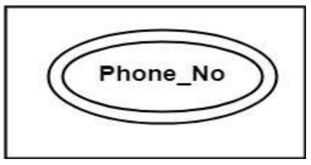


b) Composite attribute: An attribute that is a combination of other attributes is called a composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other characteristics such as pin code, state, country.

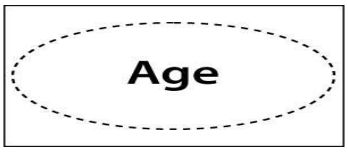


c) Single-valued attribute: Single-valued attribute contain a single value. For example, Social_Security_Number.

d) Multi-valued Attribute: If an attribute can have more than one value, it is known as a multi-valued attribute. Multi-valued attributes are depicted by the double ellipse. For example, a person can have more than one phone number, email-address, etc.



a) Derived attribute: Derived attributes are the attribute that does not exist in the physical database, but their values are derived from other attributes present in the database. For example, age can be derived from date_of_birth. In the ER diagram, Derived attributes are depicted by the dashed ellipse.



iii) Relationships: The association among entities is known as relationship. Relationships are represented by the diamond-shaped box. For example, an employee works_at a department, a student enrolls in a course. Here, Works_at and Enrolls are called relationships.

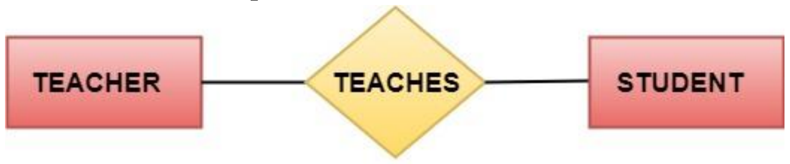
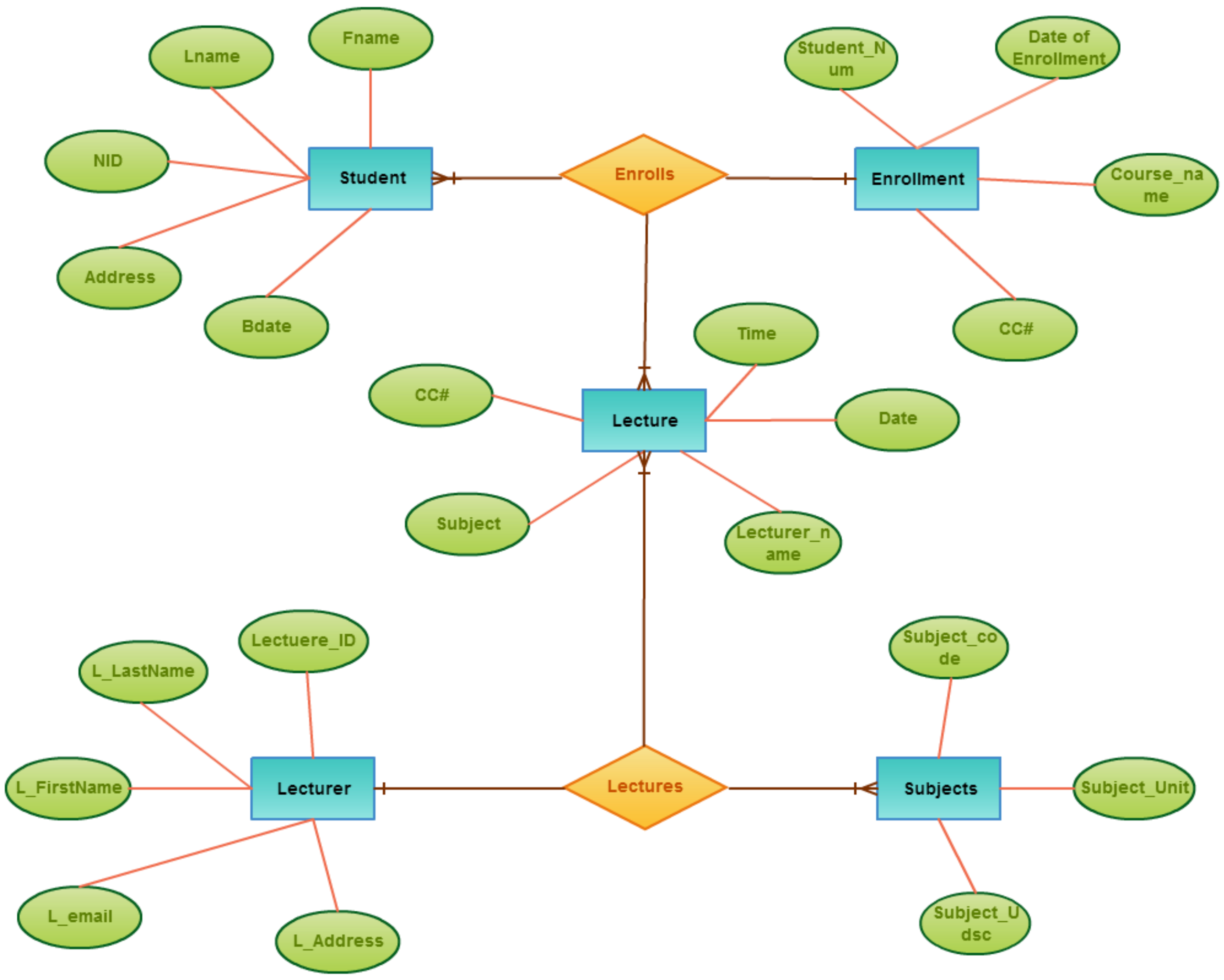


Fig: Relationships in ERD

ER DIAGRAM FOR STUDENT ENROLLMENT SYSTEM



7 Decision Table

Decision tables are used in various engineering fields to represent complex logical relationships. This testing is a very effective tool in testing the software and its requirements management. The output may be dependent on many input conditions and decision tables give a tabular view of various combinations of input conditions and these conditions are in the form of True(T) and False(F). Also, it provides a set of conditions and its corresponding actions required in the testing.

Parts of Decision Tables:

In software testing, the decision table has 4 parts which are divided into portions and are given below :

- Condition Stubs : The conditions are listed in this first upper left part of the decision table that is used to determine a particular action or set of actions.
- Action Stubs : All the possible actions are given in the first lower left portion (i.e, below condition stub) of the decision table.
- Condition Entries : In the condition entry, the values are inputted in the upper right portion of the decision table. In the condition entries part of the table, there are multiple rows and columns which are known as Rule.
- Action Entries : In the action entry, every entry has some associated action or set of actions in the lower right portion of the decision table and these values are called outputs.

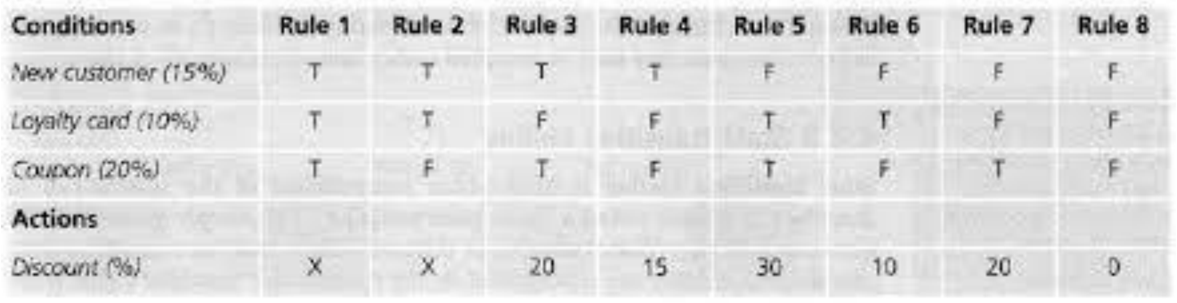
Types of Decision Tables: The decision tables are categorized into two types and these are given below:

- Limited Entry : In the limited entry decision tables, the condition entries are restricted to binary values.
- Extended Entry : In the extended entry decision table, the condition entries have more than two values. The decision tables use multiple conditions where a condition may have many possibilities instead of only 'true' and 'false' are known as extended entry decision tables.

Applicability of Decision Tables :

- The order of rule evaluation has no effect on the resulting action.
- The decision tables can be applied easily at the unit level only.
- Once a rule is satisfied and the action selected, n another rule needs to be examined.
- The restrictions do not eliminate many applications.

Example



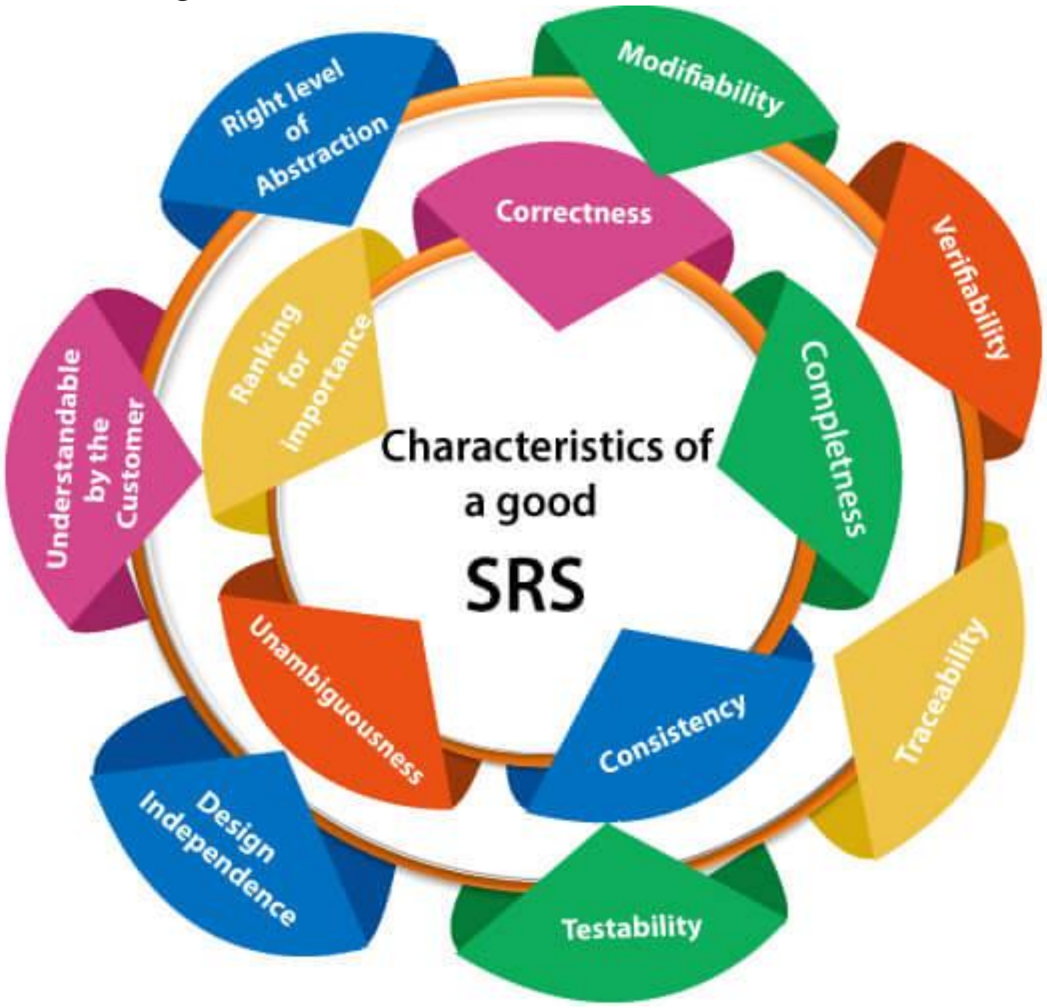
Conditions	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**	**Rule 5**	**Rule 6**	**Rule 7**	**Rule 8**
New customer (15%)	T	T	T	T	F	F	F	F
Loyalty card (10%)	T	T	F	F	T	T	F	F
Coupon (20%)	T	F	T	F	T	F	T	F
Actions								
Discount (%)	X	X	20	15	30	10	20	0

8 Software Requirement Specification

The production of the requirements stage of the software development process is Software Requirements Specifications (SRS) (also called a requirements document). This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. SRS is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.

The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system. The two methods create entirely various situations and establish different purposes for the document altogether. The first case, SRS, is used to define the needs and expectation of the users. The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.

Characteristics of good SRS



Following are the features of a good SRS document:

- Correctness
- Completeness
- Consistency
- Unambiguousness

- Ranking for importance and stability
- Modifiability
- Verifiability
- Traceability
- Design Independence
- Testability
- Understandable by the customer
- The right level of abstraction (if needed then you can defined them by yourself as they all are discussed in class)

Topic comes under SRS are-

- Introduction
    - Purpose
    - Scope
    - Definition, Acronyms, Abbreviation

- Overall Description
    - Product Perspective
    - Software Requirement
        - Front End
        - Back End
    - Hardware Requirement
    - Functional Requirement
    - Non-Functional Requirement
    - Diagrams (And so on as discussed in class….also can refer AIM-2 of SE lab file)

https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document#:~:text=A%20software%20requirements%20specification%20(SRS)%20is%20a%20document%20that%20describes,stakeholders%20(business%2C%20users).

9 Software Quality Assurance

SQA is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly. Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process. Generally, the quality of the software is verified by the third-party organization like international standard organizations.

Software Quality Assurance (SQA) encompasses
- SQA process
- Specific quality assurance and quality control tasks (including technical reviews and a multitiered testing strategy)
- Effective software engineering practice (methods and tools)
- Control of all software work products and the changes made to them

- Procedure to ensure compliance with software development standards (when applicable)
- Measurement and reporting mechanisms

Elements Of Software Quality Assurance:

- 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 all work products conform to them.
- Reviews and audits: Technical reviews are a quality control activity performed by software engineers for software engineers. Their intent is to uncover errors. Audits are a type of review performed by SQA personnel (people employed in an organization) with the intent of ensuring that quality guidelines are being followed for software engineering work.
- Testing: Software testing is a quality control function that has one primary goal—to find errors. The job of SQA is to ensure that testing is properly planned and efficiently conducted for primary goal of software.
- 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: SQA ensures that adequate change management practices 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 which is 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: The SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

Software quality assurance focuses on:
- software's portability
- software's usability
- software's reusability
- software's correctness
- software's maintainability
- software's error control

Software Quality Assurance has:
- A quality management approach
- Formal technical reviews
- Multi testing strategy
- Effective software engineering technology
- Measurement and reporting mechanism

Major Software Quality Assurance Activities:

- 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.
- 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.
- 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 check the compatibility of this fix with whole project.
- Multi testing Strategy: Do not depend on a single testing approach. When you have a lot of testing approaches available use them.
- 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.
- Managing Reports and Records: Document and share QA activities (test cases, defects, client changes) for future reference and stakeholder alignment.

Benefits of Software Quality Assurance (SQA)

- SQA produces high quality software.
- High quality application saves time and cost.
- SQA is beneficial for better reliability.
- SQA is beneficial in the condition of no maintenance for a long time.
- High quality commercial software increase market share of company.
- Improving the process of creating software.
- Improves the quality of the software.
- It cuts maintenance costs. Get the release right the first time, and your company can forget about it and move on to the next big thing. Release a product with chronic issues, and your business bogs down in a costly, time-consuming, never-ending cycle of repairs.

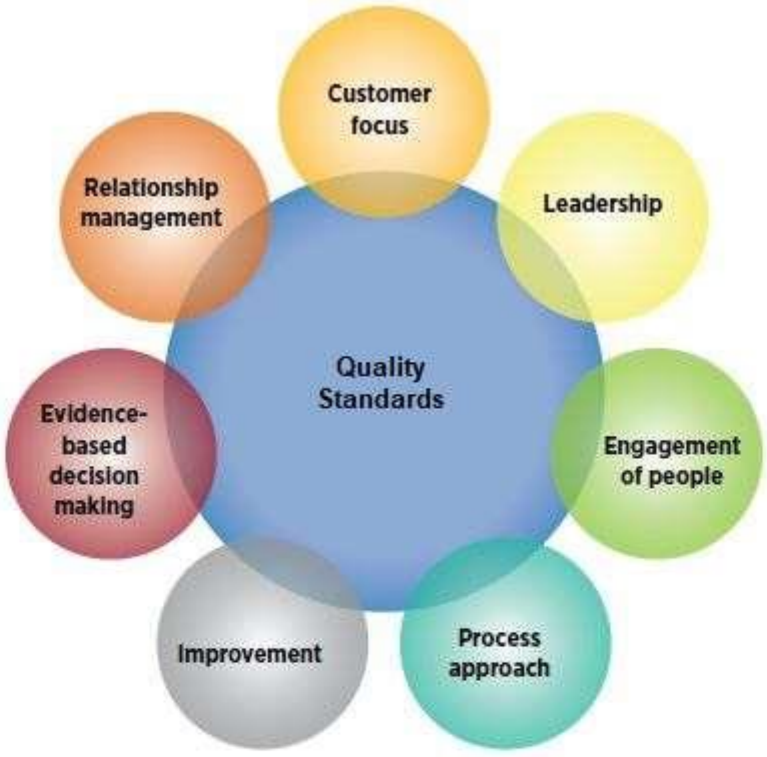
Disadvantage of SQA:

- 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.

Overview of Quality Standards/Frameworks

Quality standards are defined as documents that provide requirements, specifications, guidelines, or characteristics that can be used consistently to ensure that materials, products, processes, and services are fit for their purpose.

Standards provide organizations with the shared vision, understanding, procedures, and vocabulary needed to meet the expectations of their stakeholders. Because standards present precise descriptions and terminology, they offer an objective and authoritative basis for organizations and consumers around the world to communicate and conduct business.



Principles of Quality Standards

Organizations turn to standards for guidelines, definitions, and procedures that help them achieve objectives such as:

- Satisfying their customers' quality requirements
- Ensuring their products and services are safe
- Complying with regulations
- Meeting environmental objectives
- Protecting products against climatic or other adverse conditions
- Ensuring that internal processes are defined and controlled

10 ISO 9000

The International organization for Standardization is a world wide federation of national standard bodies. The International standards organization (ISO) is a standard which serves as a for contract between independent parties. It specifies guidelines for development of quality system. Quality system of an organization means the various activities related to its products or services. Standard of ISO addresses to both aspects i.e. operational and organizational aspects which includes responsibilities, reporting etc. An ISO 9000 standard contains set of guidelines of production process without considering product itself.

ISO 9000 Certification: There are several reasons why software industry must get an ISO certification. Some of reasons are as follows :

- This certification has become a standards for international bidding.
- It helps in designing high-quality repeatable software products.
- It emphasis need for proper documentation.
- It facilitates development of optimal processes and totally quality measurements.

Features of ISO 9001 Requirements :

- Document control –All documents concerned with the development of a software product should be properly managed and controlled.
- Planning –Proper plans should be prepared and monitored.
- Review –For effectiveness and correctness all important documents across all phases should be independently checked and reviewed .
- Testing –The product should be tested against specification.
- Organizational Aspects –Various organizational aspects should be addressed e.g., management reporting of the quality team.

Advantages of ISO 9000 Certification

- Business ISO-9000 certification forces a corporation to specialize in "how they are doing business". Each procedure and work instruction must be documented and thus becomes a springboard for continuous improvement.
- Employees morale is increased as they're asked to require control of their processes and document their work processes
- Better products and services result from continuous improvement process.
- Increased employee participation, involvement, awareness and systematic employee training are reduced problems.

Shortcomings of ISO 9000 Certification :

- ISO 9000 does not give any guideline for defining an appropriate process and does not give guarantee for high quality process.
- ISO 9000 certification process have no international accreditation agency exists.

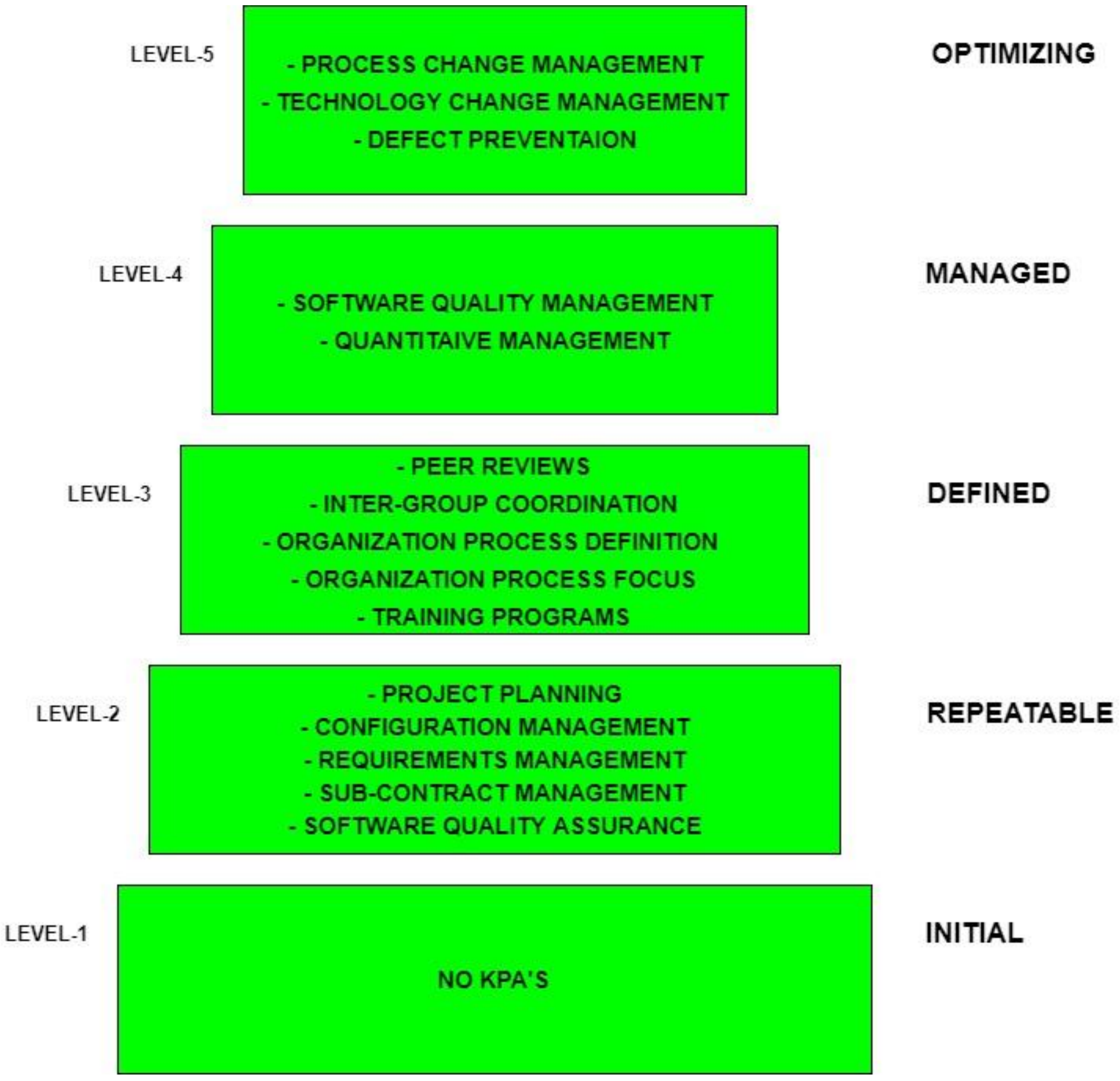
11 SEI-CMM

CMM was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in 1987.

- It is not a software process model. It is a framework that is used to analyse the approach and techniques followed by any organization to develop software products.
- It also provides guidelines to further enhance the maturity of the process used to develop those software products.
- It is based on profound feedback and development practices adopted by the most successful organizations worldwide.
- This model describes a strategy for software process improvement that should be followed by moving through 5 different levels.
- Each level of maturity shows a process capability level. All the levels except level-1 are further described by Key Process Areas (KPA's).

Each of these KPA's defines the basic requirements that should be met by a software process in order to satisfy the KPA and achieve that level of maturity.

Conceptually, key process areas form the basis for management control of the software project and establish a context in which technical methods are applied, work products like models, documents, data, reports, etc. are produced, milestones are established, quality is ensured and change is properly managed.


LEVEL-5
- PROCESS CHANGE MANAGEMENT
- TECHNOLOGY CHANGE MANAGEMENT
- DEFECT PREVENTAION
OPTIMIZING
LEVEL-4
- SOFTWARE QUALITY MANAGEMENT
- QUANTITAIVE MANAGEMENT
MANAGED
LEVEL-3
- PEER REVIEWS
- INTER-GROUP COORDINATION
- ORGANIZATION PROCESS DEFINITION
- ORGANIZATION PROCESS FOCUS
- TRAINING PROGRAMS
DEFINED
LEVEL-2
- PROJECT PLANNING
- CONFIGURATION MANAGEMENT
- REQUIREMENTS MANAGEMENT
- SUB-CONTRACT MANAGEMENT
- SOFTWARE QUALITY ASSURANCE
REPEATABLE
LEVEL-1
NO KPA'S
INITIAL

The 5 levels of CMM are as follows:

Level-1: Initial –

- No KPA's defined.
- Processes followed are Adhoc and immature and are not well defined.
- Unstable environment for software development.
- No basis for predicting product quality, time for completion, etc.

Level-2: Repeatable –

- Focuses on establishing basic project management policies.
- Experience with earlier projects is used for managing new similar natured projects.
- Project Planning- It includes defining resources required, goals, constraints, etc. for the project. It presents a detailed plan to be followed systematically for the successful completion of good quality software.

- Configuration Management- The focus is on maintaining the performance of the software product, including all its components, for the entire lifecycle.
- Requirements Management- It includes the management of customer reviews and feedback which result in some changes in the requirement set. It also consists of accommodation of those modified requirements.
- Subcontract Management- It focuses on the effective management of qualified software contractors i.e. it manages the parts of the software which are developed by third parties.
- Software Quality Assurance- It guarantees a good quality software product by following certain rules and quality standard guidelines while developing.

Level-3: Defined –

- At this level, documentation of the standard guidelines and procedures takes place.
- It is a well-defined integrated set of project-specific software engineering and management processes.
- Peer Reviews- In this method, defects are removed by using a number of review methods like walkthroughs, inspections, buddy checks, etc.
- Intergroup Coordination- It consists of planned interactions between different development teams to ensure efficient and proper fulfillment of customer needs.
- Organization Process Definition- Its key focus is on the development and maintenance of the standard development processes.
- Organization Process Focus- It includes activities and practices that should be followed to improve the process capabilities of an organization.
- Training Programs- It focuses on the enhancement of knowledge and skills of the team members including the developers and ensuring an increase in work efficiency.

Level-4: Managed –

- At this stage, quantitative quality goals are set for the organization for software products as well as software processes.
- The measurements made help the organization to predict the product and process quality within some limits defined quantitatively.
- Software Quality Management- It includes the establishment of plans and strategies to develop quantitative analysis and understanding of the product's quality.
- Quantitative Management- It focuses on controlling the project performance in a quantitative manner.

Level-5: Optimizing –

- This is the highest level of process maturity in CMM and focuses on continuous process improvement in the organization using quantitative feedback.
- Use of new tools, techniques, and evaluation of software processes is done to prevent recurrence of known defects.
- Process Change Management- Its focus is on the continuous improvement of the organization's software processes to improve productivity, quality, and cycle time for the software product.
- Technology Change Management- It consists of the identification and use of new technologies to improve product quality and decrease product development time.
- Defect Prevention- It focuses on the identification of causes of defects and prevents them from recurring in future projects by improving project-defined processes.