Module 2

Process Framework Models & Phases in Software Development

PROCESS FRAMEWORK MODELS

Ques 1) What is a Process Framework?

Ans: Process Framework

A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects, regardless of size or complexity. It also includes a set of umbrella activities that are applicable across the entire software process.

Some most applicable framework activities are shown in **figure 2.1**:

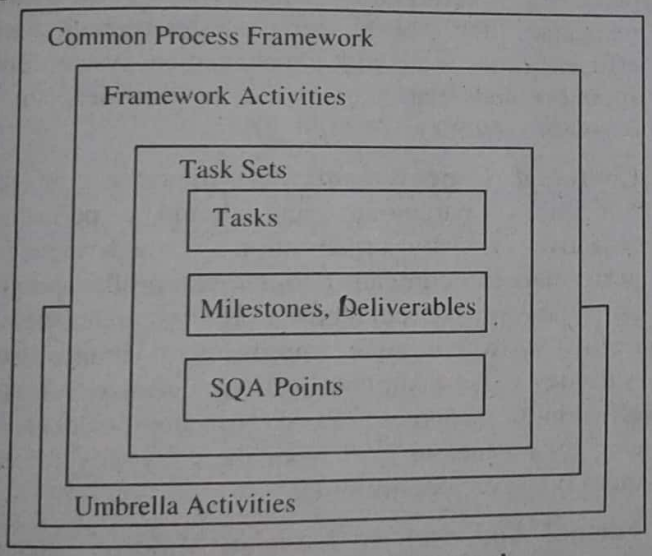

Figure 2.1: Process Framework

The five generic **process framework activities** are as follows:

1) **Communication:** This framework activity involves heavy communication and collaboration with the customer (and other stakeholders) and encompasses requirements gathering and other related activities.

2) **Planning:** This activity establishes a plan for the software engineering work that follows. It describes the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.

3) **Modelling:** This activity encompasses the creation of models that allow the developer and the customer to better understand software requirements and the design that will achieve those requirements.

testing that is required to uncover errors in the code.

5) **Deployment:** The software (as a complete entity or as a partially completed increment) is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.

These five generic framework activities can be used during the development of small programmes, the creation of large web applications, and for the engineering of large, complex computer-based systems. The details of the software process will be quite different in each case, but the framework activities remain the same.

Typical **umbrella activities** include:

1) **Software Project Tracking and Control:** It allows the software team to assess progress against the project pan and take necessary action to maintain schedule.

2) **Risk Management:** It assesses risks that may affect the outcome of the project or the quality of the product.

3) **Software Quality Assurance:** It defines and conducts the activities required to ensure software quality.

4) **Formal Technical Reviews:** It assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next action or activity.

5) **Measurement:** It defines and collects process, project, and product measures that assist the team in delivering software that meets customers' needs; can be used in conjunction with all other framework and umbrella activities.

6) **Software Configuration Management:** It manages the effects of change throughout the software process.

7) **Reusability Management:** It defines criteria for work product reuse (including software components) and establishes mechanisms to achieve reusable components.

8) **Work Product Preparation and Production:** It encompasses the activities required to create work products such as models, documents, logs, forms, and lists.

Some popular **process framework models** are as follows:

1) ISO 9000 standards, and

2) Capability Maturity Model (CMM).

Ques 2) Explain the ISO 9000 quality standards model.

Ans: ISO 9000 Quality Standards

ISO 9000 is a series of quality management system standards that have been created to help organisations ensure that they meet the needs of their stakeholders, such as customers, shareholders, personnel, etc., while at the same time complying with statutory and regulatory requirements.

The ISO 9000 series consists of the following standards:

1) **ISO 9000:2015:** It defines the specifics of quality management systems, which form the basis of the ISO 9000 family, while also stipulating the terms used in these standards.
2) **ISO 9001:2015:** This is the requirement standard which provides directions on how to achieve quality requirements, meet the relevant regulatory requirements, improve satisfaction for all stakeholders and have a method of identifying and implementing improvements.
3) **ISO 9004:2009:** It offers guidance for sustained success according to the eight quality management principles. It lays out the principles that senior management should use to improve performance while taking into account the requirements of all stakeholders.
4) **ISO 19011:2011:** It sets out guidance on internal audits, which are used to confirm and improve the effectiveness of a quality management system, and external audits, which are generally audits of suppliers but can also be done by any interested third parties.

ISO 9001 is the only auditable standard for which third party certification and auditing organisations provide independent confirmation that the requirements of that standard are met.

ISO 9000 was introduced in 1987 and was based on the BS 5750 series of standards as presented by the British Standard Institute (BSI) to the International Organisation for Standardisation (ISO) in 1979.

Ques 3) What are the ISO 9000 Principles?

Ans: Principles of the ISO 9000 Standard

1) **A Customer Focus:** As stated before, the customer is the primary focus of a business. By understanding and responding to the needs of customers, an organisation can correctly targeting key demographics and therefore increase revenue by delivering the products and services that the customer is looking for. With knowledge of customer needs, resources can be allocated appropriately and efficiently. Most importantly, a business's dedication will be recognised by the customer, creating customer loyalty. And customer loyalty is return business.
2) **Good Leadership:** A team of good leaders will establish unity and direction quickly in a business environment. Their goal is to motivate everyone working on the project, and successful leaders will minimise miscommunication within and between departments. Their role is intimately intertwined with the next ISO 9000 principle.
3) **Involvement of People:** The inclusion of everyone on a business team is critical to its success. Involvement of substance will lead to a personal investment in a project and in turn create motivated, committed workers. These people will tend towards innovation and creativity, and utilise their full abilities to complete a project. If people have a vested interest in performance, they will be eager to participate in the continual improvement that ISO 9000 facilitates.
4) **Process Approach to Quality Management:** The best results are achieved when activities and resources are managed together. This process approach to quality management can lower costs through the effective use of resources, personnel, and time. If a process is controlled as a whole, management can focus on goals that are important to the big picture, and prioritise objectives to maximise effectiveness.
5) **Management System Approach:** Combining management groups may seem like a dangerous clash of titans, but if done correctly can result in an efficient and effective management system. If leaders are dedicated to the goals of an organisation, they will aid each other to achieve improved productivity. Some results include integration and alignment of key processes. Additionally, interested parties will recognise the consistency, effectiveness, and efficiency that come with a management system. Both suppliers and customers will gain confidence in a business's abilities.
6) **Continual Improvement:** The importance of this principle is paramount, and should a permanent objective of every organisation. Through increased performance, a company can increase profits and gain an advantage over competitors. If a whole business is dedicated to continual improvement, improvement activities will be aligned, leading to faster and more efficient development. Ready for improvement and change, businesses will have the flexibility to react quickly to new opportunities.
7) **Factual Approach to Decision Making:** Effective decisions are based on the analysis and interpretation of information and data. By making informed decisions, an organisation will be more likely to make the right decision. As companies make this a habit, they will be able to demonstrate the effectiveness of past decisions. This will put confidence in current and future decisions.
8) **Supplier Relationships:** It is important to establish a mutually beneficial supplier relationship; such a relationship creates value for both parties. A supplier that recognises a mutually beneficial relationship will be quick to react when a business needs to respond to customer needs or market changes. Through close contact and interaction with a supplier, both organisations will be able to optimise resources and costs.

Ques 4) What do you understand by SEI– CMM?
Or
What are the different levels of SEI-CMM model?

Ans: SEI– CMM

Goal of the SEI is to provide leadership in the state of software engineering to improve quality of systems throughout the lifecycle. The SEI contract was competitively awarded to Carnegie Mellon University (CMU).

As an assessment tool, the CMM provides the means to measure an organization's process maturity against a set of common feature that are specified at each of the maturity levels. The premise is that the more mature the process, the more likely an organization is to produce software products on time, on budget and to the specified quality.

The SEI – CMM is also used as a guideline for software process improvement (SPI) initiatives. At each maturity level, Key Process Areas (KPA) are specified. Within each Key Process Area, activities are defined which when implemented, should result in the achievement of SPI goals. These activities can be used as a prescription for software process improvement initiatives. SEI has published empirical results that indicate significant return on investment can be achieved when using the CMM as a framework for process improvement.

The best-known CMMs are: Software Capability Maturity Model, Systems Engineering Capability Maturity Model, Newly Integrated Model, and CMMI.

The intent of the CMMs is for an organization to choose the model that best fits their company's objectives and use it both as a benchmark to determine the organization's maturity and to identify areas for process improvement. In some cases, more than one model may apply.

Software Capability Maturity Model

The Software Capability Maturity Model (SW-CMM) is a model used by organizations for appraising the maturity of their software processes and for identifying practices that will increase the maturity of those processes.

The federal government requested a method for assessing a software subcontractor's process capability in mid – 1980's. In response to this, the SEI developed a process maturity framework with assistance from Mitre Corporation. This framework was released in 1987 along with a questionnaire used to determine the process maturity of an organization. By 1991, this framework had evolved into the Capability Maturity Model (CMM) version 1.0.

The capability Maturity Model is based on the following premises:

1) Software quality of an information technology system in large, stem from quality of the processes used to create it.
2) The level of technology used in information technology systems must be appropriate to the maturity of the processes.
3) System delivery is a process that can be managed, measured and progressively improved

Level of SEI-CMM

Capability Maturity Model (CMM) defines five levels of maturity for an organization. Each of these levels focuses on a set of process goals. As an organization moves through the levels, the processes introduced build on the maturity established in the previous levels (see **figure 2.2**).

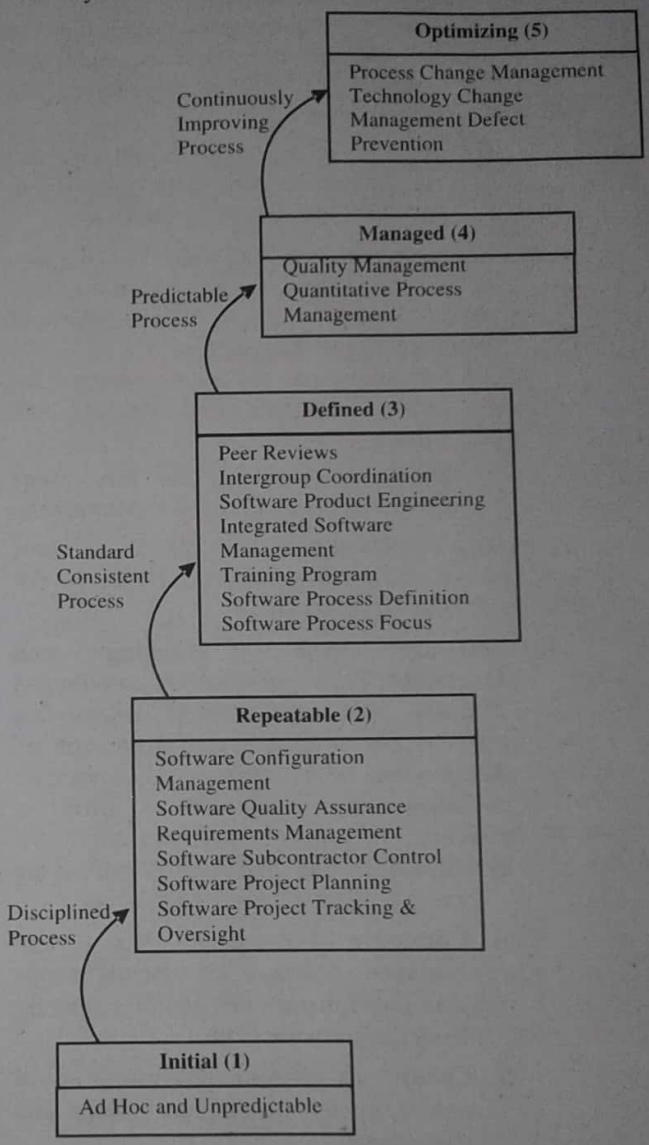

Figure 2.2: Key Process Areas

Level 1) Initial Level: This level is characterized by ad hoc activities. Very few or no processes are defined. Different engineers follow their own process and as a result the development efforts become chaotic. It is also called chaotic level. Formal project management practices are not followed.

Level 2) Repeatable Level: The goal of achieving Level 2 is to establish basic project management planning and tracking processes. This is accomplished through the six key processes areas:

i) **Requirements Management:** The purpose of Requirements Management is to establish an agreed to requirement specification that will form the basis for the project plan. The

requirement specification is agreed to by the client and other affected stakeholders. Once approved, the requirements are placed under the control of Software Configuration Management. The goals of Requirements Management are stated as:

a) System requirements allocated to software are controlled to establish a baseline for software engineering and management use

b) Software plans, products, and activities are kept consistent with the system requirements allocated to software.

ii) **Software Project Planning:** The purpose of Software Project Planning is to establish realistic project plans. The goals of Software Project Planning are:

a) Software estimates are documented for use in planning and tracking the software project

b) Software project activities and commitments are planned and document

c) Affected groups and individuals agree to their commitments related to the software project.

iii) **Software Project Tracking and Oversight:** The purpose of Software Project Tracking and Oversight is to provide management with accurate information on the progress of the project and to identify deviations from the plan as early as possible. The goals of this KPA are:

a) Actual results and performances are tracked against the software plans.

b) Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans.

c) Changes to software commitments are agreed to, by the affected groups and individuals.

iv) **Software Sub-Contract Management:** This KPA, combines the concerns of the five other Level 2 KPA's for the purpose of selecting and managing sub-contractors. There are four goals:

a) The prime contractor selects qualified software subcontractors

b) The prime contractor and the software subcontractor agree to their commitments to each other.

c) The prime contractor and the software subcontractor maintain ongoing communications.

d) The prime contractor tracks the software subcontractor's actual results and performance against its commitments.

v) **Software Quality Assurance:** The purpose of Software Quality Assurance is to monitor the software processes used on the project. This provides management with an objective view into the project. This KPA is in the support of four goals.

a) Software quality assurance activities are planned.

b) Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively.

c) Affected groups and individuals are informed of software quality assurance activities and results.

d) Non-compliance issues that cannot be resolved within the software project are addressed by senior management.

vi) **Software Configuration Management:** The purpose of Software Configuration Management is to maintain the integrity of the system's configuration items throughout the entire lifecycle of the product. The goals are:

a) Software configuration management activities are planned

b) Selected software work products are identified, controlled, and available

c) Changes to identified software work products are controlled

d) Affected groups and individuals are informed of the status and content of software baselines

Level 3) Defined Level: The Key practices at Level 3 address both project and organization issues. The focus is on the establishment of organizational level processes that are used by all projects. The project team can tailor the organizational processes to suite their specific needs.

i) **Organization Process Focus:** The purpose of the Organization Process Focus KPA is to establish the infrastructure for continuously improving the software process across the organization

ii) **Organization Process Definition:** The purpose of this KPA is to develop and maintain the organization's software processes. Information about the software process assets must be collected, maintained and readily available to the practitioners

iii) **Training Program:** The purpose of the Training Program KPA is to establish and maintain a planned organizational training program. Through the program, the practitioners receive the skills and knowledge they require to perform their software engineering roles.

iv) **Integrated Software Management:** The purpose of Integrated Software Management is to integrate the organization's management and software engineering processes into a seamless set of coherent processes.

v) **Software Product Engineering:** The purpose of Software Product Engineering is to produce correct and consistent software work products.

This includes the development, maintenance, documentation and verification of requirements, design, code, documentation, and testing.

vi) **Inter-group Co-ordination:** The purpose of Intergroup Co-ordination is to provide a formal means for the software groups to actively participate with other stakeholders departments.

vii) **Peer Reviews:** The purpose of this KPA is to plan and perform effective peer reviews to identify and remove defects as early as possible in the products life cycle.

Level 4) Managed Level: At level 4, organizations collect and analyze detailed metrics of the processes and product produced.

i) **Quantitative Process Management:** The purpose of Quantitative Process Management is to control the software processes through quantitative metrics.

ii) **Software Quality Management:** Software Quality Management involves defining software quality goals for the products and processes defining plans to achieve the goals, and monitoring the progress.

Level 5) Optimized Level: Level 5 organizations are focused on continuous improvement

i) **Defect Prevention:** The purpose of Defect Prevention is to identify causes of defects, prioritize the causes and plan the elimination.

ii) **Technology Change Management:** The purpose of Technology Change Management is to systematically introduce new technologies.

This involves the identification of new technologies, evaluation to determine value and the controlled introduction into the existing environment.

iii) **Process Change Management:** The purpose of Process Change Management is the continuous improvement of the organization wide software processes.

Ques 5) What is difference between SEI-CMM and ISO 9000 model?

Or

Compare ISO and SEI-CMM model.

Ans: Comparison between SEI-CMM and ISO 9000

Table 2.1: SEI CMM and ISO 9000

Basis	SEI-CMM	ISO 9000
Framework	Capability Maturity Model (CMM) is a five-level framework for measuring software engineering practices, as they relate to process	ISO 9000 defines a minimum level of generic attributes for a quality management program
Orientation	The orientation of SEI's CMM is the software development process.	The orientation of ISO 9000 is quality system management. In general, it is the structure and auditability of the QA function of an organization.
Usage	CMM assessment is purely for internal use	ISO 9000 is awarded by an international standards body, and thus can be quoted by an organization in official documents, in communication with customers and other parties
Development Aim	CMM was developed specifically for software industry and thus addresses several issues specific to software industry	ISO 9000 standards were basically designed to audit manufacturing/service organizations.
Objective	SEI CMM aims for achieving Total Quality Management (TQM), which is beyond quality assurance	ISO 9001 aims at Level 3 (Defined Level) of SEI CMM model.
Levels	CMM has five levels: 1) Initial 2) Repeatable 3) Defined 4) Managed 5) Optimization	ISO 9000 has no levels
Criteria	It provides grade for process maturity.	ISO 9000 provides pass or fail criteria.
Requirement	CMM gets not technical aspect of software engineering.	ISO 9000 specifies minimum requirement.
Steps	It reconnects the mechanism for step by step progress through its successive maturity levels.	ISO 9000 does not specify sequence of steps required to establish the quality system.

Elements not Included	Similarly other process in CMM are not included in ISO 9000 1) Project tracking 2) Process and technology change management 3) Intergroup coordinating to meet customer's requirements 4) Organization level process focus, process development and integrated management.	Certain process elements that are in ISO are not included in CMM like: 1) Contract management 2) Purchase and customer supplied components 3) Personal issue management 4) Packaging ,delivery, and installation management

SOFTWARE DEVELOPMENT

Ques 6) What do you understand by software development? What are the different phases in software development?

Or

Explain software development life cycle.

Ans: Software Development/Software Development Life Cycle (SDLC)

Software development organizations follow some process when developing a software product. In immature organizations, the process is usually not written down. In mature organizations, the process is in writing and is actively managed. A key component of any software development process is the lifecycle model on which the process is based. The particular lifecycle model can significantly affect overall lifecycle costs associated with a software product. Lifecycle of the software starts from concept exploration and ends at the retirement of the software.

According to IEEE standard Glossary of Software Engineering Terminology, "The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The software lifecycle typically includes a requirement phase, design phase, implementation phase, test phase, installation and check-out phase, operation and maintenance phase, and sometimes retirement phase".

A software lifecycle model is a particular abstraction that represents a software lifecycle. A software lifecycle model is often called a **Software Development Lifecycle (SDLC)**.

The Software Development Lifecycle is a process of building a good software and its lifecycle stages provides Quality and Correctness of good software. All the stages of lifecycle are important in itself. One wrong step in lifecycle can create a big mistake in the development of software.

Being an owner or stakeholder of a software company one must know the development life cycle of the software. Even buyer may also aware of this lifecycle. So everyone wants to know that how its development begins, which are the development process, which is the end portion of the development life cycle.

Phases in Software Development

We have numerous types of SDLC models like Waterfall, Agile, and Spiral etc. All this SDLC model must follow these six steps for developing errorless software. Software Development Life Cycle has totaled 6 steps. All six steps are mentioned below:

1) **Requirement Gathering and Analysis:** Requirement gathering and analysis is the first stage and major stage of any SDLC model. This phase is basically the brainstorming phase because it has the many sub stages for like feasibility analysis stages to check how much idea can put into action for development.

   In this stages, communication taking place between stakeholders, end users and the project team. So, all the person which are related to the project and they gather information for software development:
   i) Identify and capture stakeholder requirements using customer interviews and surveys
   ii) Build multiple user cases to describe each action that a user will take on the new system.

   In that a brand new software development takes place more requirement gathering process for development and in other already build software not need too much information and data gathering process.

2) **System Analysis:** This is the second phase of SDLC where the entire system is defined in detail. In fact, in this stage developer get a detailed blueprint of the various phases of the software that developed in the project.

   At this stage, the system is divided into smaller parts to make it easier more manageable for the developers, designers, testers, project managers and other professionals who are going to work on the software in the latter stages.

3) **System Design:** In this phase, the design of the system is designed. The Design is developed by the analysis and designers. The system analyst design the logical design for the designers and then designer get the basic idea of designing the software design of front and back end both.

   The system analyst and designer work together in designing the software design and designer design the best software design under the guidance of system analyst.

4) **Coding:** It is the logical part of the development process. In this phase lots of brains are working for coding and get the final successful result for the software. In this a team of programmers is assigned by the company to work on the software.

The work is subdivided under a sub-phase called Task Allocation, where each task is assigned different coder. So, the development process is working faster.

5) **Testing:** By process of coding, then the final process testing is proceeding. When the software is ready it is sent to the testing department where Quality Analysts test it thoroughly for different errors by forming various test cases.

   Once the testing department and Quality analyst makes sure that the software is error-free, then it goes to the next stage. So, the testing process is complete when all the testing module is complete.

6) **Implementation:** This the final phase of the software development life cycle. In this stage, if the software runs on various systems by users or buyers. If it runs smoothly on these systems without any flaw,then it is considered ready to be launched.

   If it generates error then it goes to testing department for testing and many coders write a new code for developers errorless software.

Ques 7) What is requirement engineering?

Ans: Requirement Engineering

Requirement engineering is the sub-discipline of software engineering that is concerned with determining the goals, functions, and constraints of software systems and the representation of these aspects in forms amenable to modeling and analysis.

Its goal is to create a requirements specification that is complete, correct, and understandable to both customers and developers. This last goal creates somewhat of a dilemma, as it indicates the duality of purpose of requirements documents to provide insight for the customers to ensure the product under development meets their needs and expectations and as a complete representation of the functions and constraints of the system as a basis for developers.

In the real-time system domain this is further complicated by the need to represent timing and performance constraints as well as the more readily elicited functional requirements.

Ques 8) What are the requirements? What are the different types of requirements? List them.

Ans: Requirements

Requirements provides the appropriate mechanism for understanding what the customer wants, analysing need, assessing feasibility, negotiating a reasonable solution specifying the solution unambiguously, validating the specification, and managing the requirements as they are transformed into an operational system.

Requirement defines the capability that the software system solution must deliver and the intended results that must result on its application to business problems. In order to generate such requirements a systematic approach is necessary, through a formal management process called as requirements management.

Types of Requirements

Some of the problems that arise during the requirements engineering process are a result of failing to make a clear separation between these different levels of description. The requirements can be distinguished by using the terms:

1) **System Requirements:** Detailed description of what the system should do.
2) **User Requirements:** High-level abstract requirements.

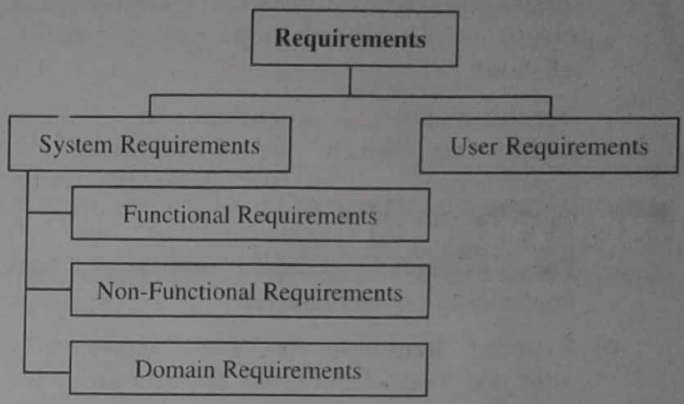

Figure 2.3: Types of Requirements

Ques 9) What are system requirements? Discuss about the different types of system requirements?

Or

Describe the Functional, Non-Functional Requirements and Domain Requirements.

Ans: System Requirements

System requirements explain the system's functions, services and operational constraints in detail. The system requirements document (sometimes called a functional specification) should be precise.

Types of System Requirements

Software system requirements are often classified as functional requirements, non-functional requirements or domain requirements:

1) **Functional Requirements:** Functional requirements should clearly state all system functionality as per customer statement of service. How a system should respond to particular inputs or how a specific system should respond in a specific situation should be clearly defined.

   The functional requirements are greatly dependent on the type of software, expected users and the type of system. Functional user requirements can be high-level statements of what the system should be doing whereas functional system requirements define in detail the system services.

2) **Non-Functional Requirements:** The non-functional requirements define system properties and constraints. Various properties of a system can be: Reliability, response time, storage requirements. And constraints

of the system can be: Input and output device capability, system representations, etc. Process requirements may also specify programming language or development method.

Non-functional requirements are more critical than functional requirements. If the non-functional requirements do not meet then the complete system is of no use.

Types of Non-Functional Requirements

The classification of non-functional requirements is given below:

i) **Product Requirements:** These requirements specify how a delivered product should behave in a particular way. **For example,** execution speed, reliability.

ii) **Organizational Requirements:** The requirements which are consequences of organizational policies and procedures come under this category.

For example, process standards used implementation requirements.

iii) **External Requirements:** These requirements arise due to the factors that are external to the system and its development process. **For example,** interoperability requirements, legislative requirements.

3) **Domain Requirements:** Functional or non-functional requirements that arise from the application domain of the system characterising the domain (e.g. physical laws, regulatory standards) are referred to as domain requirements. The necessities are forced on the system by the system's operational domain.

For example, a train control system needs to consider the braking characteristics in various weather conditions.

The characteristics and features of the system have to be defined which will redirect the domain. It may be possible that the specific requirements or new functional requirements, constraints on current necessities are described. If domain requirements are not met, the system might not work.

Ques 10) What is user requirement? Discuss the problems faced in expression of user requirements specification.

Ans: User Requirements

The user requirements should describe functional and non-functional requirements in such a way that they are understandable by system users who do not have detailed technical knowledge. User requirements are defined using natural language tables and diagrams because these are the representations that can be understood by all users.

The readers of the user requirements are not usually concerned with how the system will be implemented and may be managers who are not interested in the detailed facilities of the system. The readers of the user requirements include:

1) Client Managers
2) System End Users
3) Client Engineers
4) Contract Managers
5) System Architects

Problems in Expression of User Requirements Specification

Various problems that can arise in the requirement specifications when requirements are given in natural language:

1) **Lack of Clarity:** Sometimes requirements are given in ambiguous manner. It is expected that text should help in clear and precise understanding of the requirements.
2) **Requirements Confusion:** There may be confusion in functional requirements and non-functional requirements, system goals and design information.
3) **Requirements Mixture:** There may be a chance of specifying several requirements together as a single requirement.

Ques 11) What is requirement analysis?

Ans: Requirements Analysis

Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, analysing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Requirement analysis is a software engineering task that bridges the gap between system level requirements engineering and software design. Requirements engineering activities result in the specification of software's operational characteristics, indicate software's interface with other system elements, and establish constraints that software must meet. Requirement analysis allows the software engineer to refine domains that will be treated by software.

Software requirements analysis may be divided into five areas of effort:

1) Problem recognition,
2) Evaluation and synthesis,
3) Modeling,
4) Specification, and
5) Review

The analyst studies the system specification and the software Project Plan. It is important to understand software in a system context and to review the software scope that was used to generate planning estimates. Problem evaluation and solution synthesis is the next major area of effort for analysis.

The analyst must define all externally observable data objects, evaluate the flow and content of information, define and elaborate all software functions, understand software behavior in the context of events that affect the system, establish system interface characteristics, and uncover additional design constraints.

Throughout evaluation and solution synthesis, the analyst's primary focus is on "what" not "how". What data does the system produce and consume, what functions must the system perform, what behavior does the system exhibit, what interfaces are defined and what constraints apply?

Ques 12) What are the different steps of requirements analysis?

Ans: Steps in Requirements Analysis
The different steps of requirement analysis are depicted in **figure 2.4**.

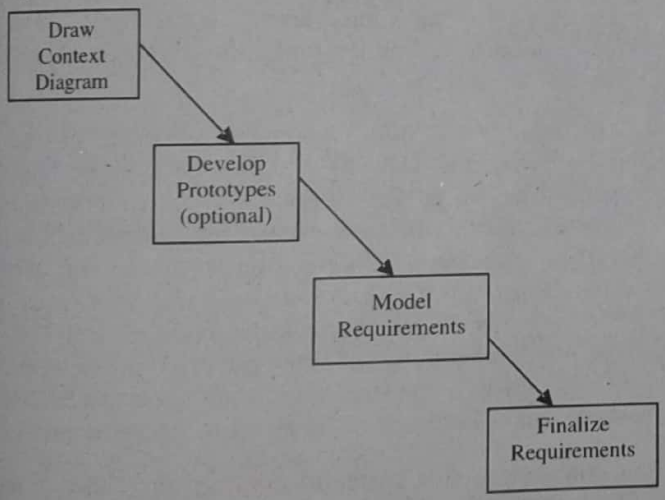

Figure 2.4: Steps in Requirements Analysis

Step 1: Draw Context Diagram: A context diagram establishes the entities that are outside the system but have close interactions with the components inside the system.

Step 2: Development of a Prototype (Optional): One of the most effective ways to understand the customer's need is building a prototype. This is a model which acts like the desired system or product. This prototype can be modified continuously as per the requirement of the customer until he is satisfied with the result. Prototypes help the customer get a touch and feel of the proposed system.

A prototype is built quickly at low cost and will hence have limitations that might be unacceptable in the final product.

Step 3: Model Requirements: This comprises of graphical representation in the form of data flow diagrams, entity relationship diagrams, data dictionaries, state-transition diagrams, etc., to understand the inter-relationships among the different functions, data and external entities. This helps in finding out incorrect, inconsistent, omitted and unnecessary requirements.

Step 4: Finalise Requirements: Modelling provides a better understanding of the system behaviour. Inconsistencies and ambiguities are identified and corrected, data flow among different modules is analysed and elicitation and analysis provides better insight into the system. After all this, the necessities are concluded and documented in a prescribed format.

Ques 13) What is requirement elicitation for software? What are the different activities of requirement elicitation process?

Ans: Requirements Elicitation
Requirements elicitation is the most difficult aspect of software development as it is a communication sensitive feature of software development. This can only succeed through an affective customer-developer partnership.

The requirements of users reside inside their minds. Hence, it is very important to understand what the users really want which can be identified from the expectations of users from the new software.

Requirements Elicitation Process Activities

1) **Requirements Discovery:** In this process, the requirements are gathered after interacting with stakeholders. Domain necessities from stakeholders are gathered and documented.
2) **Requirements Classification and Organisation:** In this stage, the requirements are classified according to similarity and inter-relation and then organised into rational clusters.
3) **Requirements Prioritisation and Negotiation:** Where multiple stakeholders are involved, conflicts will arise with respect to requirements. At this stage, requirements are prioritised and conflicts are resolved through negotiation.
4) **Requirements Documentation:** Requirements are documented for the next stage of development. This documentation can be formal or informal.

Ques 14) What are the different requirements elicitation methods?
Or
Write the methods of requirements elicitation.

Ans: Requirements Elicitation Methods
Information collecting tools are decided by the analyst and its usage. There are no standard rules on how the tools are to be used but the information needs to be acquired accurately, methodically under right conditions and with minimum interruption in the user's work. The following

figure 2.5 depicts the several methods of requirements collection:

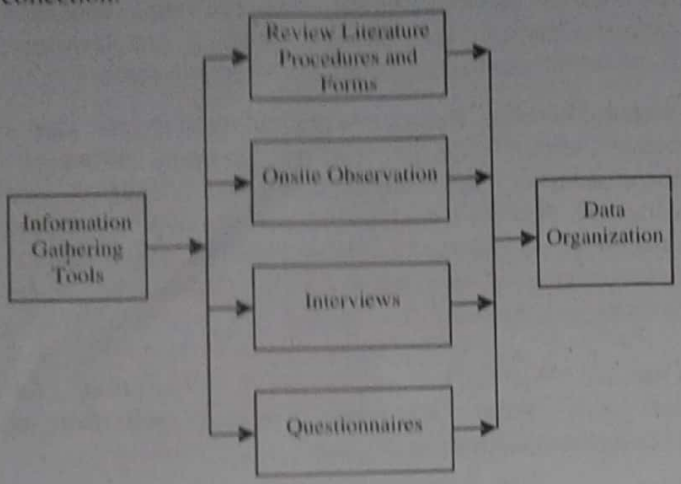

Figure 2.5: Information Gathering Tools

1) **Reviews of Literature, Procedures and Forms/Document Analysis:** Significant approaches of information collection are researching of literature on professional reference and procedures, manuals, textbooks, company studies, government publications or consultant studies. Significant approaches of information collection are researching of literature on professional reference and procedures, manuals, textbooks, company studies, government publications or consultant studies. In the document analysis phase, the main emphasis is on the documents which have been produced by the software engineers. The program codes (having the comments) and a separate document (which describes the software system) are included in these documents.

   Developers can use these collected data in re-engineering efforts (such as subsystem identification). Other documentation sources such as memos, group e-mail lists, local newsgroups and documents defining the development process, are also analysed.

2) **Onsite Observation:** Onsite observation is another tool for requirements gathering. In this, people, objects and occurrences are recognised and noted for attaining information. An analyst is a neutral person who is detached from the system being observed.

   Onsite observation is a fact finding technique. In this, the system analyst learns about the system, either by watching the person involved in performing the activities or by participating himself.

   One of the most effective tools that an analyst has is 'on-site observation'. In this, he/she finds out the ways of functioning of the system by personally visiting the sites. The role of an analyst is of an information seeker as he/she gains the knowledge of the activities, operations, and processes of the system by observing the activities himself/herself.

   The flow of documents, the users of the system, working of the current system, etc., are examined and understood by the analyst as he/she visits the organisation. The reason behind performing this task by the analyst himself/herself is that, he/she knows every minute detail about the points to be noticed and highlighted. In the development of new system, the things which can cause delay the development and are unimportant, such as, wrong requirement, are minutely observed by the analyst.

   The information gathered by the analyst himself is unbiased and hence, very meaningful. The analyst can better understand the system as he/she focuses on the real circumstances and requirements rather than those which have been documented earlier. Instead of concluding from small samples of observation, the analyst should be more focussed and he should be more patient as this technique of information gathering is time-consuming. Yet, in knowing feelings, perceptions and motivations of the people, this method cannot be effective.

3) **Interviews:** Interviews are one-to-one interaction between the observer and the interviewee. In this, the interviewer or observer asks questions to get ideas about the problem where he/she is also in a position to observe the problem first hand. As such, this is a flexible method of eliciting information.

   The technique/method of obtaining information directly from the people is 'personal interview'. Through personal interviews the analysts gain information regarding the problems, expectations and the existing system.

   An interview is the face-to-face meeting of the interviewer (the analyst) and the interviewee. The analyst has a golden opportunity to learn several relevant facts while observing an interview. The analyst can also observe the reactions of the interviewee to the questions of the interviewer. Through this observation, the interviewer is able to get the authenticity of the gathered information. Alongwith this, the questions which were not in the scene, may also be asked by assuring the interviewee.

   Many times, it is better to conduct interviews in a row, one followed by another to obtain the critical study facts. Within an organisation, the interview should be conducted at each level (i.e., from the president/chief officer to the mail clerk). The adjustment to the usual environmental variables by the analyst decides how good the interview was.

   The general questions of this process are listed in the table below:

Table 2.2: Questions

1)	Are your requirements being fulfilled by this report?
2)	Give your suggestions to enhance its quality.
3)	Which job are you carrying out?
4)	Give the objective of the job which you are carrying out.
5)	Is the information provided enough to meet these objectives?
6)	List all the extra information you need, if any?

4) **Questionnaires:** This is a tool where users are given a set of questions to which they need to answer. These questions may be closed or open-ended.

   A questionnaire gathers information, in written form, from people by asking a set of questions in a structured prescribed format. It is normally used for gathering specific information.

   Questionnaires are especially helpful when respondents are geographically scattered and they have no time to hold interviews. Thus, it provides a quick means of gathering information and analysis.

   Objectives of Questionnaires
   i) It helps to get agreement between end-users.
   ii) For deep study of organisation, it recognises the direction of area.
   iii) A post-implementation audit is to be scheduled.
   iv) The precise but changing requirements of the business operations should be identified.

Ques 15) What are the requirement analysis principles?

Ans: Requirements Analysis Principles
Requirements analysis is very important and essential activity after elicitation. One analyses, refine and study the gathered requirements in order to make consistent and unambiguous requirements.

This activity reviews all requirements and may provide a graphical view of the entire system. After the completion of analysis, it is expected that the understandability of the project may improve significantly.

Here, we may also interact with the customer to clarify points of confusion and to understand which requirements are more important than others.

Problem analysis is done to obtain a clear understanding of the needs of the clients and the users, and what exactly is desired from the software.

Some common analysis principles are as follows:
1) **Model the Data Domain:** First operational analysis principle requires an examination of the information domain and the creation of data model.
   i) Define Data Objects
   ii) Describe Data Attributes
   iii) Establish Data Relationships
2) **Model Function:** The functions that the software is to perform must be defined.
   i) Identify functions that transform data objects
   ii) Indicate how data flow through the system
   iii) Represent producers and consumers of data
3) **Model Behavior:** The behavior (dynamic often as a consequence of external events) must be represented.
   i) Indicate different states of the system
   ii) Specify events that cause the system to change state
4) **Partition the Models:** The models that depict information function and behavior must be partitioned in a manner that uncovers detail in a layered (or hierarchical) fashion.
   i) Refine each model to represent lower levels of abstraction
   ii) Refine data objects
   iii) Create a functional hierarchy
   iv) Represent behavior at different levels of detail
5) **Essence:** The analysis process should move from essential information toward implementation detail. We should begin by focusing on the essence of the problem without regard to implementation details.

Ques 16) What is software prototyping? Also discuss the prototyping methods and tools?

Ans: Software Prototyping
Software prototyping is the activity of creating prototypes of software applications, i.e., incomplete versions of the software program being developed. A prototype typically simulates only a few aspects of, and may be completely different from, the final product.

Prototyping is the technique of constructing a partial implementation of a system so that customers, users, or developers can learn more about a problem or a solution to that problem. It is a partial implementation because if it were full implementation, it would be the system, not a prototype of it.

It allows users to explore and criticise proposed systems before undergoing the cost of a full-scale development.

Prototyping Methods and Tools
There are three classes of methods and tools:
1) **Fourth Generation Techniques:** It encompasses a broad array of database query and reporting languages, program and application generators, and other very high-level nonprocedural languages.
2) **Reusable Software Components:** Other approach to rapid prototyping is to assemble, rather than build, the prototype by using a set of existing software components.
3) **Formal Specification and Prototyping Environments:** A number of formal specification languages and tools have been developed as a replacement for natural language specification techniques. Today, developers of these formal languages are in the process of developing interactive environments that:
   i) Enable an analyst to interactively create language-based specifications of a system or software,
   ii) Invoke automated tools that translate the language-based specifications into executable code, and
   iii) Enable the customer to use the prototype executable code to refine formal requirements.

Ques 17) What do you understand by specification? Discuss the principles of specification.

Ans: Specification
Requirement specification is the process of writing down the user and system requirements in a requirements document. The requirements specification phase includes the main activity of writing the Software Requirements Specification (SRS).

This document is prepared by a customer or in conjunction with a customer through requirements elicitation, i.e., the process of determining the customer's needs.

Software requirements may be specified in a variety of ways. Guidelines for representation are as follows:
1) Representation format and content should be relevant to the problem
2) Information contained within the specification should be nested
3) Diagrams and other notational forms should be restricted in number and consistent in use.
4) Representations should be revisable

Specification Principles
A number of specification principles can be proposed:
1) Separate functionality from implementation
2) Develop a model of the desired behavior of a system
3) Establish the context in which software operates
4) Define the environment in which the system operates
5) Create a cognitive model rather than a design or implementation model
6) Specification is an abstract model of a real system
7) Establish the content and structure of a specification (easy to be changed)

Ques 18) What is SRS? What are the main components of SRS document?

Ans: Software Requirements Specification (SRS)
Requirements Documentation is very important activity after the requirements elicitation and analysis. This is the way to represent requirements in a consistent format. Requirements document is called **Software Requirements Specification** (SRS).

A specification can be a written document, a graphical model, a formal mathematical model, a collection of usage scenarios, a prototype, or any combination of these.

System Specification is the final work product produced by the system and requirements engineer. It serves as the foundation for hardware engineering, software engineering, database engineering and human engineering.

It describes the function and performance of a computer-based system and the constraints that will govern its development. The specification bounds each allocated system element. The System Specification also describes the information (data and control) that is input to and output from the system.

The final output of this phase is Software Requirement Specification Document (SRS). The primary objective of analysis is problem understanding, while the basic objective of the requirements phase is to produce the SRS. During analysis modeling, performance constraints, design constraints, standards compliance, recovery etc. are not included in the model, but must be specified clearly in the SRS because the designer must know about these to properly design the system.

Components of SRS Document
SRS document comprises of a complete information description, a detailed functional description, a representation of system behavior, an indication of performance requirements and design constraints, validation criteria and other information pertinent to requirements.

Basic **issues** addressed by SRS are as follows:
1) **Functionality:** SRS must specify the functional requirements, which governs which outputs should be produced with given inputs.
2) **Performance:** SRS must specify the performance constraints on the software system. All requirements pertaining to performance characteristics must be clearly specified.
3) **Design Constraints:** SRS must specify any resource limits, operating environment, reliability and security requirements etc.
4) **External Interfaces:** SRS must specify all the possible interactions of the software with people, hardware or other software.

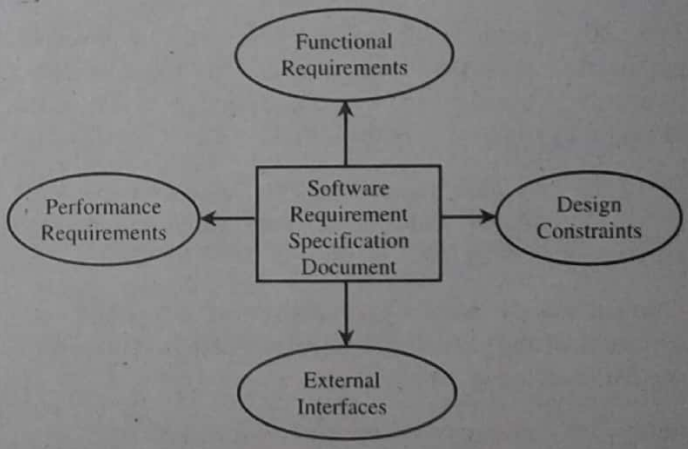

Figure 2.6: Components of SRS Document

Ques 19) What are the characteristics of good SRS document?

Ans: Characteristics of Good SRS Document
1) **Complete**
   i) SRS should be complete.
   ii) SRS defines precisely all the live situations that will be encountered and the system's capability to successfully address them.
2) **Consistent**
   i) SRS should be consistent.
   ii) SRS capability functions and performance levels are compatible, and the required quality features (security, reliability, etc.) do not negate those capability functions.

3) **Accurate**
   i) SRS precisely defines the system's capability in a real-world environment, as well as how it interfaces and interacts with it.
   ii) This aspect of requirements is a significant problem area for many SRSs.

4) **Modifiable**: The logical, hierarchical structure of the SRS should facilitate any necessary modifications and that too with a greater ease.

5) **Ranked:** Individual requirements of an SRS are hierarchically arranged according to stability, security, perceived ease/difficult of implementation, or other parameter that helps in the design of that and subsequent documents.

6) **Testable:** SRS must be stated in such a manner that unambiguous assessment criteria can be derived from the SRS itself.

7) **Traceable:** Each requirement in SRS must be uniquely identified to a source (for example, use case, government requirement, industry standard, etc.)

8) **Unambiguous**
   i) SRS must contain requirements statements that can be interpreted in one way only i.e., it should be unambiguous.
   ii) This is another area that creates significant problems for SRS development because of the use of natural language.

9) **Valid**
   i) A valid SRS is one in which all parties and project participants can understand, analyse, accept, or approve it.
   ii) This is one of the main reasons SRSs are written using natural language.

10) **Verifiable**
   i) A verifiable SRS is consistent from one level of abstraction to another.
   ii) Most attributes of a specification are subjective and a conclusive assessment of quality requires a technical review by domain experts.
   iii) Using indicators of strength and weakness provide some evidence that preferred attributes are or are not present.

Ques 20) Write the IEEE standards for SRS.

Ans: IEEE Standards for SRS

The following SRS outline is based on IEEE:

1. **Introduction:** This section is intended to provide an overview of the rest of the specification.
   - **1.1 Purpose:** This section must describe the purpose of the SRS and the intended audience.
   - **1.2 Scope:** This section must identify the product, explain what the product will and will not do, and describe the application of the software, including benefits, objectives, and goals.
   - **1.3 Definitions:** This section must identify all terms, acronyms, and abbreviations used in the specification.
   - **1.3 References:** This section must identify all documents referenced elsewhere in the specification.
   - **1.4 Overview:** This section must describe what the rest of document contains.
2. **Overall Description:** This section is intended to provide the background to understand the rest of the requirements:
   - **2.1 Product Perspective:** This section must put the product into perspective with other products. It will usually include a block diagram of the larger system. It should specify constraints (e.g., system interfaces with other software), user interfaces (e.g., screen formats, timing), hardware interfaces, software interfaces (e.g., versions of interfaced software), memory constraints, operations (e.g., modes of operations), and site adaptation constraints.
   - **2.2 Product Functions:** This section must include a summary of the major functions of the product.
   - **2.3 User Characteristics:** This section must include the educational level, experience, and technical expertise of the users.
   - **2.4 Constraints:** This section must include any other (e.g., regulatory) constraint that is not covered in product perspective.
   - **2.5 Assumptions and Dependencies:** This section must include any assumptions that, if not true, would require changes to the requirements.
   - **2.6 Apportioning of Requirements:** This section must identify requirements that may be delayed to future versions of the product.
3. **Specific Requirements:** According to IEEE Standard 830: "This section of the SRS should contain all the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements". This is an important criteria to remember: The SRS should be sufficiently detailed so designs and tests can be constructed directly from the SRS. Also, according to IEEE Standard 830: "These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output".
   - **3.1 External Interface Requirements:** This section must describe all inputs and outputs of the system. This is detailing the information from product perspective.
   - **3.2 Functions:** This section must describe all the functions of the system. These must include validity checks on inputs, responses to abnormal situations, effect of parameters, and the relationship of outputs to inputs.
   - **3.3 Performance Requirements:** This section must describe the static and dynamic requirements.
   - **3.4 Design Constraints:** This section must describe any constraints on the design.