Chapter 1
Introduction to Software Engineering

Software:

Software is the collection of computer programs, procedure rules and associated documentation and data.

Classification of software:

There are two types a) System software b) Application software

System software are Operation system, Assemblers, Debuggers, Compilers, Utilities and File magnitude, etc.

Application software are Image processors, Databases, Games, Word processors, Spreadsheets and Communication software, etc.

Characteristic of Software:

a) Maintainability: This is a critical attribute because software change is inevitable consequence of a changing business environment.

b) Dependability: Dependability including reliability, security and safety. Dependable software should not cause any physical or economic damage in the event of system failure.

c) Efficiency: Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency includes responsiveness, processing time, memory utilization etc.

d) Usability: Software must be usable. This means that it should have an appropriate user interface and adequate documentation.

e) Software is intangible: The product cannot be touched to get an idea of its quality.

f) Software is developed. It is not manufactured.

g) Software does not wear-out.

Introduction to Software Engineering:

Definition 1: The systematic approach to the development, operation and maintenance of a software product.

Definition 2: Software engineering is the application of science and mathematics by which the capabilities of a computer are made useful to man via computer programs procedures and associated documentation. (Barry Boehm)

Components of Software Engineering:

a) Software Development Life Cycle (SDLC):
It defines the various stages and activities associated with the development of a software system.

b) Software Quality Assurance (SQA):
This is the process of ensuring user satisfaction through the development of a quality product.

c) Software Project Management (SPM):
It is the application of the principles project management to the process of software development.

d) Software Management(SM):
They are the methods and procedures to be followed for effective maintenance and change control.

e) Computer Aided Software Engineering (CASE):
They are set of automated tools that support the process of software development.

Goals of Software Engineering:

a) The goal of software engineering is to create practical software system that have social and/or economic value using a systematic software development process.

b) To improve the efficiency and quality of software production in order to cope for complex, secure, distributed, real time, adaptable, dependable and so on.

Software Product:

Definition: Software products are software systems that are delivered to a customer with a documentations which describes how to install and use the system.

There are two types of software products a) Generic products b) Customized products

Generic Product	Customized Product
Generic products are developed for anonymous customer.	The customized products are developed for specific customers.
The target is generally the entire world and many copies are expected to be sold.	The specific product is designed and developed as per customer requirements
Examples: operating systems, compilers, analyzers, word processors, CASE tools etc.	Examples: Payroll System, Inventory System etc.

Software Development Life Cycle (SDLC)

Definition: **SDLC** is sequence of activities carried out by analyst, designer, and user to develop and implement an information system. These activities are carried out in different stages.

Analyst: Analyst is the one who studies the requirements of the customer and define problem domain.

Designer: Designer is the one who design the system in terms of structure of databases, screens, forms, and reports. She/he also determines the hardware and software requirements for the system to be developed.

User: User is the one who uses the system.

The entire software development life cycle can be broadly classified into seven phases:

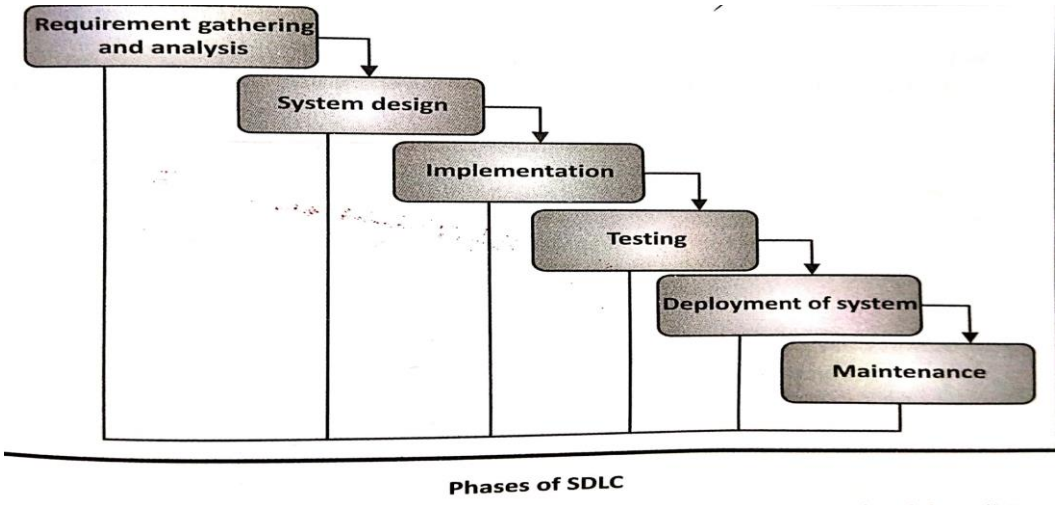
1) Feasibility Study 2) Requirement Analysis 3) System Design 4) Development 5) Testing 6) Implementation 7) Software Maintenance

Feasibility Study: In this phase, we assist whether or not a project should be undertaken. This stage involves defining the problem and fixing up its boundaries. At the end of this stage, the design and development team becomes clear with the project objectives and their work preview.

Requirement Analysis: In this stage, the requirements are studied and analysed. The technical development team works with customers and system end users to identify the application domain, functions, services performance capabilities, hardware constraints related to the system to be developed.

System Design: In this phase, a new system is designed according to the needs of the user. It is the phase which find a solution for the given problem. This is the phase where the specification of each and every component of the project are considered.

Development: This is the phase where the system is actually developed. The whole of design phase is built and implemented in this phase.



Testing: During this phase entire project functionality is tested with all of its units integrated as a whole and then the system is tested with test data. During this phase the developed system is reviewed against each and every customer requirement specification. The developed system should be able to address all of user's needs and its functionality.

Implementation: This is the process in which the developed system is handed over to the client. The old system is dispensed, new system is put into operation for use and all the users are trained to manage and maintain the new system.

Software Maintenance: This is the phase wherein the development team maintains the system for the client. It includes adding enhancements, improvements, updates to the newer and not just correction of errors and emergency fixed of a system break down.

Different types of SDLC Models

1) Waterfall Model
2) Prototype Model
3) Rapid application development model (RAD)
4) Evolutionary Development
5) Incremental Model
6) Iterative Model
7) Component based Software Engineering
8) Spiral Model

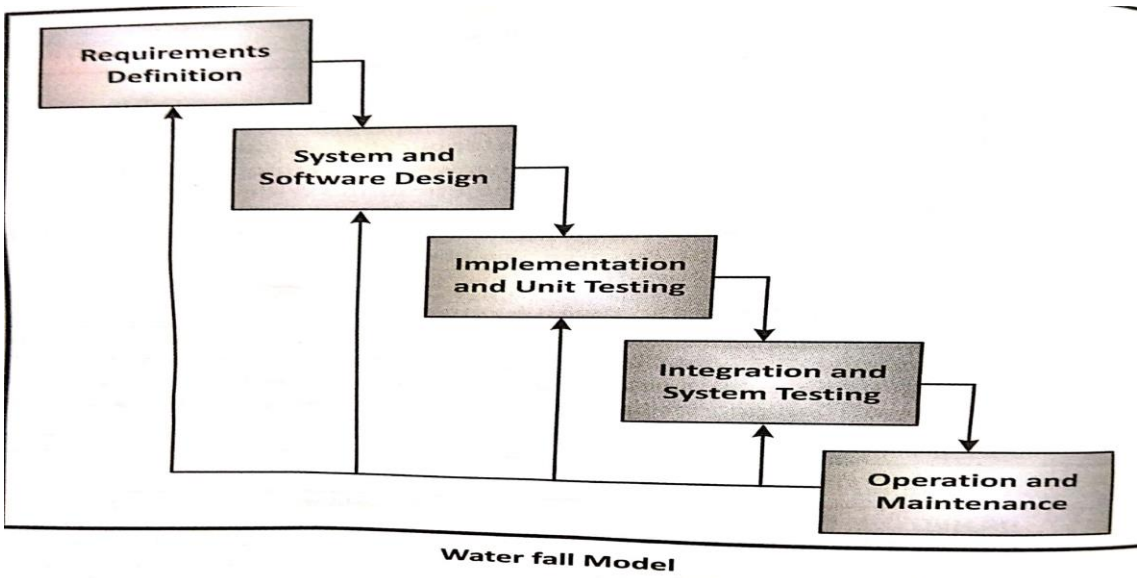
Waterfall Model:

The phases of waterfall model are

a) Requirement Definition
b) Software and System Design
c) Implementation and Unit Testing
d) Integration and System Testing
e) Operation and Maintenance

Requirement analysis and definition:

During this phase the requirements of the customer is properly understood and documented. The objective of this phase is to document all functions, performance and constraints of the software.



System and software designing:

System design process partitions the requirements to either hardware or software systems. System and software designing includes architectural design. Abstractions and relationships are design in this phase.

Implementation and unit testing:

Software must be implemented and tested to meet the specific requirements.

Integration and system testing:

In this phase, the programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, system is delivered to the customer.

Operation and Maintenance:

It involves maintenance of the software and keep the software operational after it delivers to the client.

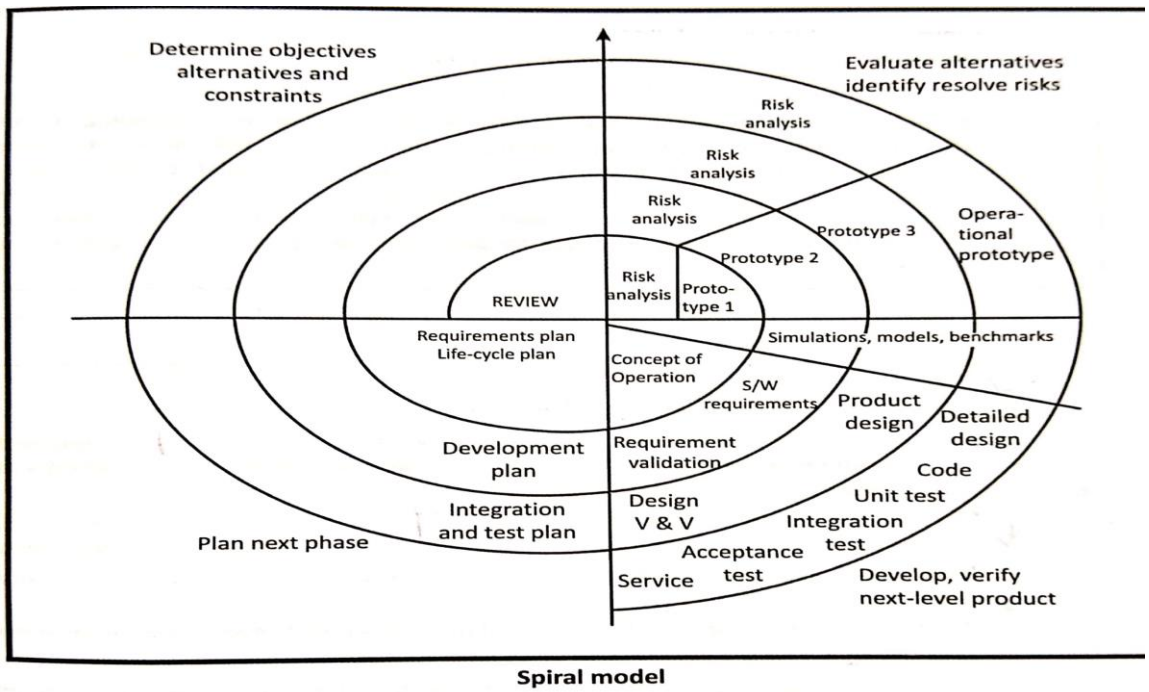
Advantages:

a) It is simple in nature.
b) It provides systematic approach.
c) It is easy to maintain.
d) It provides clarity to software engineers of what they need to do.
e) This model underlines the need for discipline, planning and management in the process of software development.

Disadvantages:

a) It is difficult to define all requirements at the beginning of the project.
b) Model is not suitable for accommodating any change.
c) Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
d) Few business system have stable requirements.
e) The waterfall model is mostly used for large systems engineering projects where a system is developed at several stages.
f) This model is only appropriate when the requirements are well understood and changes will be fairly limited during the design process.

Spiral Model (Iterative Model):



Spiral model

The spiral life cycle model is a type of iterative software development model which is generally implemented in high risk projects. In this system development model, it combines the best features of both, waterfall model and prototype model. Each loop might in the spiral represents a phase of the software process. The innermost loop might be concerned with system feasibility, the next loop with requirements definition, the next loop with system design and so on.

Each loop in the spiral is split into four sectors:

a) **Objective Setting**: Specific objective for that phase of the project are defined. Constraints on the process and the product are identified and a detailed management plan is drawn up. Project risk are identified.

b) **Risk Assessment and risk reduction**: For each of the identified project risks, a detailed analysis is carried out. Steps are taken to reduce the risk.

c) **Development and validation**: After risk evaluation, a development model for the system is chosen.

d) **Planning**: The project is reviewed and a decision made whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.

Advantages:

1) High amount of risk analysis.
2) Good for large and mission-critical projects.
3) Spiral Life Cycle Model is one of the most flexible SDLC models in place.
4) Project monitoring is very easy and effective.
5) Risk management is one of the in-built features of the model, which makes it extra attractive compared to other models.
6) It is suitable for high risk projects, where business needs may be unstable.
7) A highly customized product can be developed using this.

Disadvantages:

1) Cost involved in this model is usually high.
2) Risk analysis requires highly specific expertise.
3) Project's success is highly dependent on the risk analysis phase.
4) Doesn't work well for smaller projects.
5) It is not suitable for low risk projects.

Software engineering challenges:

The software engineering challenges are

a) The Legacy Challenge
b) The Heterogeneity Challenge
c) The Delivery Challenge
d) The Trust Challenge

Legacy Challenge:

Most of the huge software systems in use today were developed many years ago. But still they are performing critical business functions. This is the legacy challenge which is the challenge of maintaining and updating the software.

Heterogeneity Challenge:

Now a day's software systems have to operate as distributed systems, across global networks that include different types of computers working on different kinds of support systems. Sometimes it is required to integrate new system with the older systems written in different programming languages.

The heterogeneity challenge therefore is the challenge of developing techniques to build dependable software which is flexible enough to fulfil the varied requirements.

Delivery Challenge:

Many software engineering techniques for software development are time consuming since they require enough time to achieve software quality. However, in today's world business must be fast, responsive and change rapidly. Accordingly, the supporting software must also change rapidly.

Therefore the delivery challenge is the challenge of delivering the software systems in a short span of time, without compromising on system quality.

Trust challenge:

It is important that the customer / user must trust the software especially when it is accessed through a web page or web service interface. Therefore, the trust challenge is to develop the software in such a way that it is trusted by its users.

Risk Management:

Risk: Risk is anything that may affect the achievement of an organization's objectives.

Risk Management: Risk Management is the process involved with identifying, analysing and responding to risk. It includes maximizing the results of positive risks and minimizing the consequences of negative events.

The process of risk management involves the following important stages:

a) Risk Identification
b) Risk Analysis
c) Risk Planning
d) Risk Monitoring

Risk Identification: Risk identification is the first step in risk assessment, which identifies all the

possible project, product and business risks in a particular project.

Risk Analysis: The next step is to analyse them by assessing the probability of the undesirable event occurring and the loss that will occur if that event occurs.

Risk Planning: Risk identification and analysis should be done during project planning. By performing risk assessment, a properly prioritized list of the project risks will become available and therefore plans of avoiding or minimizing the effects of the risks could be made.

Risk Monitoring: Risk monitoring must be a continuous process. Each identified risk is regularly assessed to check whether or not the risk is becoming more or less probable.

Difference between Software Engineering and System Engineering

Software Engineering	System Engineering
1. Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system.	1. System engineering is concerned with all aspects of computer based systems development including hardware, software and process engineering.
2. Software engineering is based on computer science, information science and discrete mathematics whereas traditional engineering is based on mathematics, science and empirical knowledge.	2. System engineers are involved in system specification, architectural design, integration and development.
3. In software engineering two main concerns are cost of development and reliability measured by the no. of errors per thousand lines of source code.	3. In traditional engineering, two main concerns for a product are cost of production and reliability measured by time to failure.

Chapter – 2
System Engineering

System:

A system is a collection of inter related components that work together achieve some objective.

System Engineering:

System Engineering is the activity of specifying designing, implementing, validating, installing and maintaining system as a whole.

System Procurement:

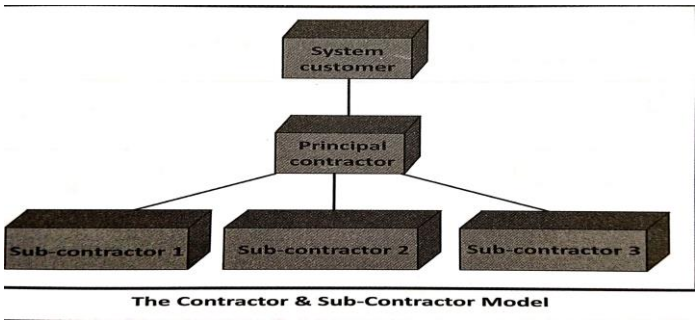
System procurement is the process of acquiring a system for an organisation to meet some identified need. The system may me brought as a whole, may be brought as a separate parts that are then integrated or may be specially designed and developed. It may be necessary to complete some system specification and architectural design before procurement decisions are made. There are two reason for this.

a) To buy or let a contractor to design and built a system a high level specification of what that system should do must be completed.

b) The specification may allow you to buy a commercial off the shelf (COTS) system. It is always almost cheaper to buy a system rather than developing a system from scratch.

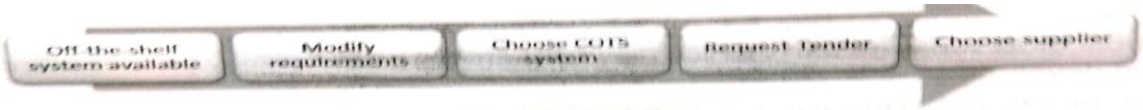
- **The Model of System Procurement:**

a) **The contractor and sub-contractor model**: The contractor and sub-contractor model minimizes the number of organisation which the procurer must deal with. The sub-contractor design and built parts of the system to a specifications produced by the principal contractor. Once completed these different parts are integrated by the principal contractor. They are then delivered to the customer by buying the system. The procurement of large hardware or software system is usually based around some principal contractor. Depending on the contract the procurer may allow the principal contractor a free choice of sub-contractor or may require principal contractor to choice sub-contractor from an approved list.

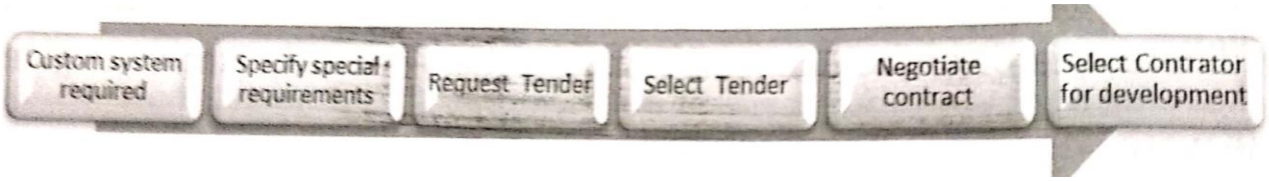


The Contractor & Sub-Contractor Model

b) **Commercially-Off–The–Shelf (COTS) System**: Normally COTS components do not match the system requirements exactly. Therefore a software engineer will have to choose an off-the-shelf system which is the closest match between the system requirements and the facilities offered by COTS system. Requirements may have to be modified to match the capabilities of off-the-shelf components.



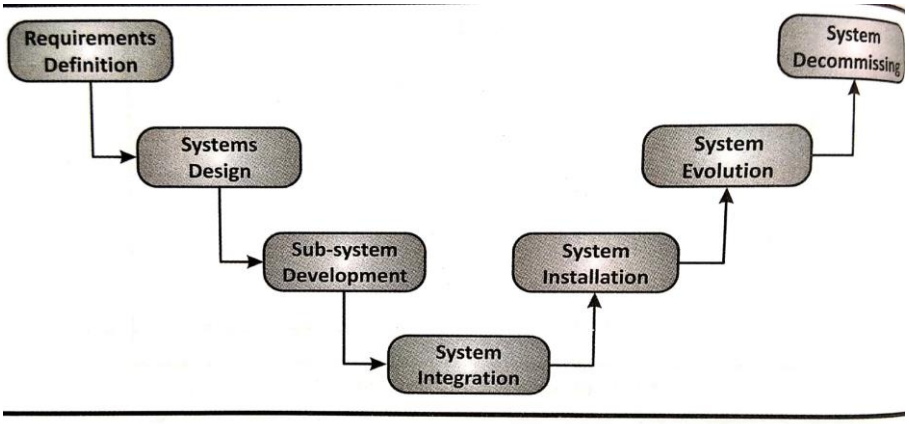
c) **Custom System**: In the case of customized system, the requirements specification must be specified in the contract document. The requirements specification may be part of the contract for the development of the system. There is usually a contract negotiation period after the contractor has been selected to build a system. During this period changes to requirements and cost changes may be agreed.



System Engineering Process:

The phases of system engineering process are

a) Systems requirements definition
b) System design
c) Sub-system development
d) System integration
e) System installation
f) System operation
g) System evolution
h) System decommissioning



System Engineering Process

a) System requirements definition:

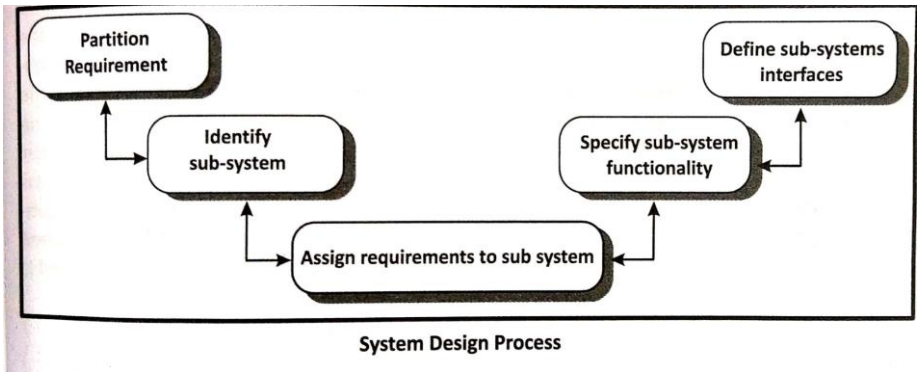
The system requirements definition process involves describing the requirements for the system as a whole, after consulting system customers and end-users. The requirements phase defines the following types of requirements.

1) **Abstract functional requirements**:
   →The basic system functions are defined in an abstract way.
   →Details of functional requirements are given at the sub-system level.
2) **Non - functional requirements**:
   →The non-functional system properties are defined which include properties such as availability, performance, safety etc.
   →These properties affect the requirements of the integrated system which includes all the sub-system.
3) **Shall - not characteristics:**
   →What the system must not do must be specified i.e. the unacceptable system behaviour must be specified.

b) System design process:

System design is concerned with how the system functionality is to be provided by the components of the system. The activities involved in this process are

a) Partition requirements
b) Identify sub-systems
c) Assign requirements to sub-system
d) Specify sub-system functionality
e) Define sub-system interfaces.



System Design Process

a) **Partition requirements**: Analyse the requirement and organize them into related groups.
b) **Identity sub-system**: The different subsystem required are identified.
c) **Assign requirements to sub-systems**: The requirements specified are assigned to the sub-systems. However, there may be problems matching the requirements with externally purchased sub-system (COTS). This may lead to modifications in the requirements.
d) **Specify sub-system functionality and inter-relationships:** The specific functioning of each sub-system are specified. The inter-relationship between sub-systems must also be specified.
e) **Define sub-system interfaces:** This activity defined the interfaces required by each sub-system. It is a critical activity for parallel development of the sub-system.

c) Sub-System Development:

The different subsystems identified during system design process are implemented. There are two option in the development process.

-→ Develop all sub-systems from scratch.

-→ Buy commercial off-the shelf systems (COTS).

When the **sub-system** is a software system, a software process involves Requirements, Design, Implementation and validation.

The advantages of buying **COTS** sub-systems and integrating them into the system are that they are cheaper, faster and tested.

d) System Integration:

System integration involves taking independently sub-systems and putting them together to make up a complete system. Integration can be done in two ways.

➔ Big bang method: All the sub-system are integrated at the same time.

➔ Incremental integration: Sub systems are integrated one at a time. Incremental integration is the best approach for two reasons.

a) It is usually impossible to schedule the development of all the sub-systems so that they are all finished at the same time.
b) Incremental integration reduces the cost of error location.

Once the components have been integrated, system testing takes place. This testing should be aimed at testing the interfaces between components and the behaviour of a system as a whole.

e) System installation:

System installation is the activity of installing the system in the environment in which it is intended to operate, while this may appear to be a simple process, there are many different problems that the installation of a complex system can take many months or even years.

Some of the problems that can arise during system installation are

→Operator training has to be identified.

➔ There may be physical installation problem.

➔ Environment assumption may be incorrect.

➔ May be human resistance to the introduction of new system.

➔ system may have to co-exist with an existing systems for some time.

f) System Evolution:

Large and complex systems have a very long life time. It is inevitable that during their life time some system evolution will take place. This may be due to errors in the original system requirements or to meet changing requirements.

Evolution is inherently very costly for a number of reasons.

➔ Changes must be analysed and approved by a range of people.

➔ Changes of one sub-system may affect other sub-systems. Consequently changes to these sub-system may be required.

➔ People in charge of system evolution need to find out why particular decisions were made.

➔ Cost of maintaining a system increases with time. System structure becomes corrupted because of continuous maintenance changes.

g) System Decommissioning:

System decommissioning involves taking the system out of service, after its useful lifetime.

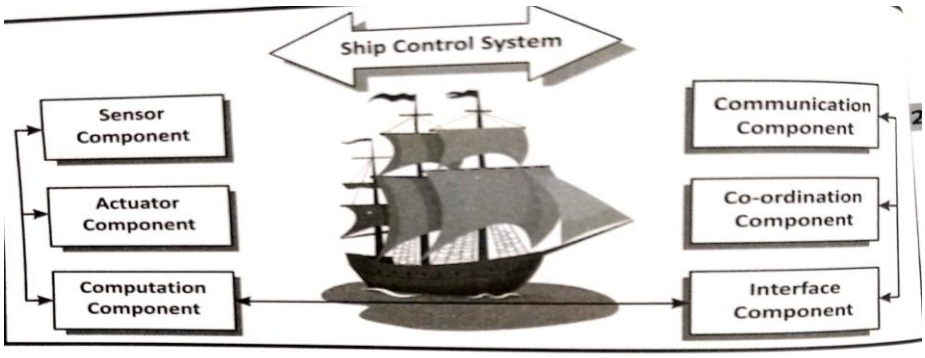
➔ Physical hardware decommissioning problems are, may require removal of materials which can pollute the environment. Should be planned during system design.

➔ When software is concerned, there is no physical decommissioning issues. However, it may be required to restructure data in the system and convert it, to be used in some other system.

System Architecture modelling

An architectural model generally presents an abstract view of sub-systems making up the system. The system architecture model is normally presented as a black diagram showing the major sub-systems and the inter-connection between these sub-systems.

The following diagram represent the architecture of a ship controller system.



Architecture model of Ship control

Function components of ship control.

1) Sensor components.
2) Actuator components.
3) Computation components.
4) Communication components.
5) Co-ordination components.
6) Interface components.

1) **Sensor components**: Collect information from a system environment about the nautical miles covered by ships, tides and waves and geography of oceans.
2) **Actuator components**: causes some change in system environment to suit the performance.
3) **Computation components**: components which perform computation by accepting the input, process the input to give the required output are called computation components.
4) **Communication components**: Establishes communication contact between one ship to another ship on the high seas during emergency times.
5) **Co-ordination components**: It is very useful to co-ordinate between the crew members of ship to steer the ship in right direction.
6) **Interface components:** Interface components transform the representation used by one system components into the representation used by other components.

Human Factor

Almost all systems have human users and therefore appropriate user interface is critical for successful system operation. Other human factors which must be taken into account by system engineers include:

a) Process Changes: Whenever the system requires changes to the work process, training will have to be organized. However, if the changes are significant or involve employees losing their jobs, then the workers may offer stiff resistance.
b) Job changes: When a new system is installed, the users may have to change the way they work. However any design that makes the managers to change their way of working is often resisted and resented.
c) Organisational changes: The political power structure in an organisation may be influenced by the new system.

Systems Reliability Engineering

System reliability is a complex concept which is always considered at the system level rather than at individual component level. Since components in a system are inter-dependent, faults in one component may affect the operation of other components.

The overall system reliability is a function of

a) Hardware reliability
b) Software reliability
c) Operator reliability

i) Hardware reliability: Hardware reliability depends on the probability of a hardware component failing and the time required to repair that component.

ii) Software reliability: Since software does not wear out, software failure is different from hardware failure. Hardware cannot be used, once it is worn out whereas software can continue to operate even after a wrong output has been produced. Therefore, software faults that occur in exceptional situations have little effect on the system's reliability.

iii) Operator reliability: Operator errors occur under stress when system failures are occurring. The operator errors may increase the stress on the hardware, causing more failures. Therefore, through the sub-system failure could have been recovered, it may develop into a serious problem leading to complete system shutdown.

Chapter - 3
Software Requirement Analysis and Specification

Software requirement specification:

A structured document setting out detailed description of the system services written as a contract between client and contractor.

Software specification:

A detailed software description which can serve as a basic for a design or implementation written for developers.

Types of requirement:

a) **Functional requirements**: Statements of services that the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
b) **Non-Functional requirements**: Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
c) **User requirements:** Statements in natural language and diagrams of the services that the system provides and its operational constraints. It is written for customers.
d) **System requirements:** A structured document setting out detailed descriptions of the system services. Written as a contract between client and contractor.

Functional Requirements:

Statements of services that the system should provide, how the system should react to particular inputs and how the system should behave in particular situations

Functional requirements specify what the product must do in order to satisfy the basic reason for its existence. They are

→Specifications of the product's functionality
→Actions that the product must take – check, compute, record, and retrieve.
→Derived from the basic purpose of the product.
→Normally business-oriented, rather than technical.
→Not the technical solution constraints that are often referred as the system requirements.
→Derived mostly from the use case scenarios.
→Not a quality.
→To be free from ambiguities.

Non-functional requirements (NFR):

Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

There are three types i) Product Requirements ii) Organizational Requirements iii) External Requirements.

Product Requirements: Requirements which specify that the delivered product must behave in a particular way, e.g. execution speed, reliability etc.

Organizational Requirements: Requirements which are a consequence of organizational policies and procedures, e.g. process standards used, implementation requirements etc.

External Requirements: Requirements which arise from factors which are external to the system and its development process, e.g. interoperability requirements, legislative requirements etc.

Software Requirement Specification Document (SRS)

SRS is a perfect detailed description of the behaviour of the system to be developed. The SRS document is an formal agreement between the developer and the customer covering the functional and non-functional requirements of the software to be developed.

Components of SRS:

a) Functionality
b) Environment Description and System Objectives
c) Project Management
d) System delivery and installation Requirements
e) Functional constraints
f) Design Constraints

Functionality:

The functional requirements include description of

→Procedures for starting up and closing down the system.
→Self-test procedures.
→Operation under normal conditions.
→Operation under abnormal conditions.
→Procedures for controlling the mode of operation.
→Recovery procedures.
→Procedures for continuing under reduced functionality.

Environment Description and System Objectives:

→Physical attributes of the environment: size, shape, and locality.
→Organizational attributes: office applications, military applications.
→Models of potential users
→Safety/security/hazards

Project Management:

Life cycle requirements: How system development will proceed (system documentation, standards, procedures for model, testing and integration, procedures for controlling change, assumptions / expected changes).

System Delivery and Installation Requirements

These requirements are: Deliverables, deadlines, acceptance criteria, quality assurance, document structure/standards/training/manuals/support and maintenance.

Functional Constraints:

They describe the necessary properties of the system behaviour described in the functional requirements. Examples performance, efficiency, response times, safety, security, reliability, quality, and dependability.

Design Constraints:

The user may want that the software satisfy certain additional conditions. These conditions are hardware and software standards, particular libraries and operating systems to be used, and compatibility issues.

Data and Communication Protocol Requirements:

They are inputs, outputs, interfaces, and communication protocols between system and environment.

Structure of SRS Document

IEEE suggests the following structure for requirements document.

1. Introduction
   1.1 Purpose of the requirement's document.
   1.2 Scope of the product.
   1.3 Definitions, acronyms and abbreviations.
   1.4 References to supporting documents.
   1.5 Overview of rest of SRS.
2. General Description
   2.1 Product perspective.
   2.2 Product functions.
   2.3 User characteristics.
   2.4 General constraints.
   2.5 Assumptions and dependencies.
3. Functional Requirements.
4. Non-functional Requirements
5. System Architecture
6. System Models
7. Appendices

1. Introduction
   1.1. **Purpose**: Identify the purpose of this SRS.
   1.2. **Scope of the System specified**: Describe the origin of the need for this system.
   1.3. **Definitions, Acronyms, and Abbreviations**: Provide the definitions of all the terms, acronyms, and abbreviations required to properly interpret the SRS.
   1.4. References to supporting Documents
   1.5. **Overview of rest of SRS**: Describe the rest of the SRS and how it is organized.
2. **General Description**
   2.1. **Product perspective**: Describe the relationship of software to its environment
   2.2. **Product Functions**: Provide a summary of all the functions of the software.
   2.3. **User characteristics**: Provide general characteristics of eventual users (for example, educational background and amount of product training).
   2.4. **General constraints**: Express the hardware limitations, interfaces, and implementation language requirements.
   2.5. **Assumptions and Dependencies**: List and describe each of the factors that affect the requirements and SRS.
3. **Functional Requirements**: The functional requirements are going to be written in narrative form identified with numbers.
4. **Non-functional Requirements**: The same format as the functional requirements is to be used for the non-functional requirements.
5. **System Architecture**: This section presents a high-level overview of the anticipated system architecture using a class diagram.
6. **System Model**: This section presents the use case diagram for the system under development.
7. **Appendices**
   7.1 Appendix A.
   Data dictionary: Completely defined entries for all of the actors and use cases in the use case diagram.

7.2. Appendix B.

Raw use case point analysis: This appendix contains the actor summary table and use case summary table for the raw use case point analysis for use case diagram.

7.3. Appendix C.

Screens and reports with navigation matrix: This appendix contains the screens and reports with their navigation matrix for each use case in the use case diagram.

7.4. **Appendix D.**

Scenario analysis tables: This appendix contains the scenario analysis table for each use case in the use case diagram.

7.5. **Appendix E.**

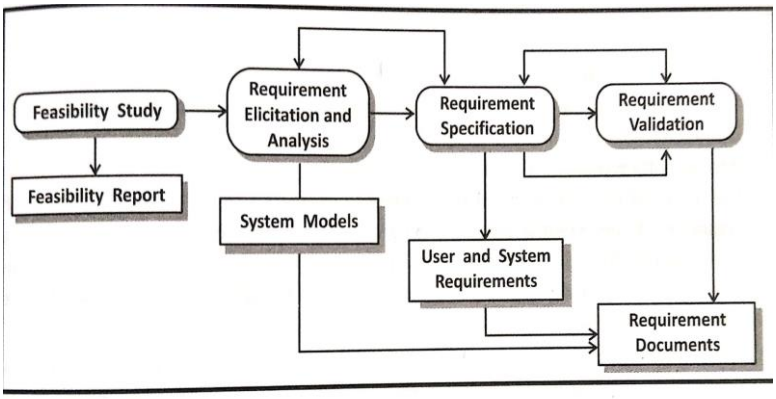
Screens/reports list

Requirement Engineering Process

The process of establishing the services that the customer requires from a system and the constraints under which it operates is called requirement engineering process.

The activities involved in the requirement engineering include.

- ➔ Feasibility study.
- ➔ Requirement elicitation and analysis.
- ➔ Requirement validation.
- ➔ Requirement management.
- ➔ Requirement documents.



Requirement Engineering process

Feasibility study:

A feasibility study looks at the viability of an idea with an emphasis on identifying potential problems and attempts to answer one main question: will the idea work and should we proceed with it?

A feasibility study is a focussed study of the following:

→**Organizational feasibility**: The system should contribute to the overall objectives of the organization.

→**Technical feasibility**: The proposed system must be developed within given cost and schedule constraints.

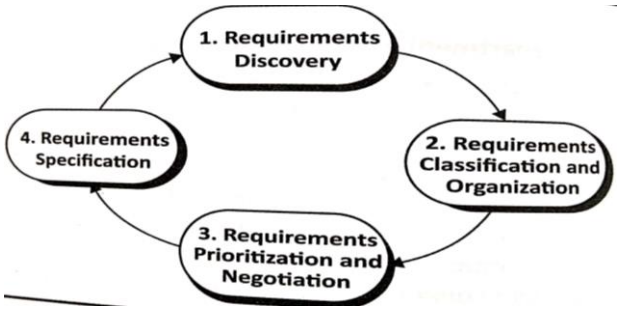
→**Operational feasibility**: It should be possible to integrate the new system with other existing systems.

Requirement Elicitation and Analysis:

The requirements elicitation and analysis process involves a wide variety of people known as stakeholders. Stakeholders include any person or group who is affected by the system, end-users who interact with the system, engineers who are developing or maintaining the systems, business managers, domain experts and trade union representatives.

The activities involved in requirement elicitation and analysis include

- ➔ Requirements discovery
- ➔ Requirements classification and organization
- ➔ Requirements prioritization and negotiation
- ➔ Requirements specification



Requirements discovery:

This is the process of interacting with stakeholders of the system to discover their requirements. Domain requirements from stakeholders and documentation are also discovered during this activity.

Requirements classification and organization:

This is an activity that takes unstructured collection of requirements, groups them into related requirements and organises them into clusters.

Requirements negotiation and prioritization:

When multiple stakeholders are involved, requirement conflicts will arise. This activity therefore involves finding and resolving conflicts in the requirements through negotiation. Further it also involves interacting with stakeholders to find out the most important requirements and prioritise them.

Requirements specification:

The requirements are documented and input into the next round of the spiral. Formal or informal requirements documents may be produced.

Requirement specification

Requirement specification adds on further information to the requirements definition. Specification is presented with the system models, useful for designing the software. It includes all the specification and constraints on the operations of the system. There are five different specification methods.

a) Structured Natural Language: Defining standard form or template to express the requirement specification using natural language.
b) Design Description: This method concentrates on operations and constraints of the system. It makes use of programming languages with more abstract features.
c) Requirement Specification Language: Various special purpose languages have been designed to express software requirement such as PSL/PSA (Problem Statement Language / Problem Statement Analyser).
d) Graphical Notations: Best known graphical notation for requirement is SADT (Structured Analysis and Design Technique).
e) Mathematical Specification: These are notations based on formal mathematical concept such as finite machines or more basic concept such as sets.

Requirements Validation

The requirements in the requirements document must be checked during the validation process. These checks are as below:

Validity checks: This process checks whether the system provides the functions that support the customer's needs in the best possible way.

Consistency checks: Requirements in the document should not conflict. This process checks for requirements conflicts.
Completeness checks: The document is checked to see if all functions and constraints defined by the customer are included.
Realism checks: The requirements are checked to ensure that they can be implemented with the given available budget and technology.
Verifiability: System requirements must be written such that they are verifiable. This means that a set of tests must be written that can verify that the delivered system meets al specified requirements.
Requirements Validation Techniques:
There are many validation techniques:

a) Requirement Reviews
b) Prototyping
c) Test case generation

Requirement Reviews: Review may also checks for

- ➔ Verifiability: Is the requirement as stated realistically testable.
- ➔ Comprehensibility: Is the requirement properly understandable by the procurers or end users of the system.
- ➔ Traceability: The origin of a requirement should be traceable. Traceability enables to understand the impact of changes on the entire system.
- ➔ Adaptability: can the requirement be changed without large scale effects on the system requirements.

Prototyping: In this approach an executable model of the system is demonstrated to end users and customers. They can experiment with this model to see if its meet their real needs.
Test case generation: The requirements engineers may need to generate specific test case to identify certain errors which other validation techniques may not detect.

Volatile Requirements

Volatile requirements are unstable requirements and are likely to change during the system development process or after the system has been put into (operational) use.
For example:

- ➔ In Hospital Management System, requirements resulting from government health care policies, innovative surgeries, new sophisticated surgical equipment etc.
- ➔ In college management system requirement resulting as changes in the university rules and regulations, pattern of the examination and up gradation in the course syllabus etc.

Types of volatile Requirement:

a) **Mutable Requirements**: Requirements that changes due the system environment in which the organization is operating.
b) **Emergent Requirements**: Requirement that emerge as the customers understanding of the system develops during the system development.
c) **Consequential Requirements**: Requirement that result from the introduction of the computer system. Introducing the computer system may change the organization processes and open up new ways of working.
d) **Compatibility Requirements**: Requirements that depend on other systems or business process within an organization.

System Models

System models are graphical representation that describes business processes, the problem to be solved and the system that is to be developed.

Types of models are

a) **Context Model**: is an architectural model which describes environment and boundary of the system.
b) **Behavioural Model**: The internal functioning and inter-relationship between various level and processes of the proposed systems. There are two types of behavioural models. **Dataflow models:** which model the data processing in the system. **State machine models**: which model how the system react to events.
c) **Sate Machine Model / Event Model:** This model describes how a system responds to internal or external events.
d) **Semantic Models:** Semantic data models describe the logical structure of the data which is imported to and exported by the systems.
e) **Object Models:** Object models that are developed during requirements analysis may be used to represent both systems data and its processing. Therefore they combine some of the uses of data-flow and semantic data models. They are also useful for showing how entities in the system may be classified and composed of other entities. There are three types of models:
   - ➔ Inheritance models.
   - ➔ Object aggregation model.
   - ➔ Object behaviour model.

Inheritance model describes the process of inheriting/ acquiring properties and behaviour from one class to another class.

Object embedded within other object is called as **object aggregation**.

The object behaviour model indicates behaviour of individual objects and overall behaviour of the object oriented system.