Introduction to System Engineering

Chapter 1: Introduction to Systems and System Life Cycle

Definition

- **System**:
    - An organized set of interrelated components working together toward some common objective.
    - It is an integrated composite of people, products and processes that provide a capability to satisfy a stated need or objective.
    - It is a collection of different elements that interact to produce results that are not obtainable by the elements alone.
    - In systems engineering, defines a system as a combination of interacting elements organized to achieve one or more stated purposes.
- system contains:
    - Components
    - Interconnections
    - Purpose or Goal
    - Boundaries
    - Inputs and Outputs
    - Control and Feedback

Types of system

- Physical/Conceptual systems
- Open systems
- Closed systems
- Natural and Human made systems
- Precedented and Unprecedented systems

System and it's Environment

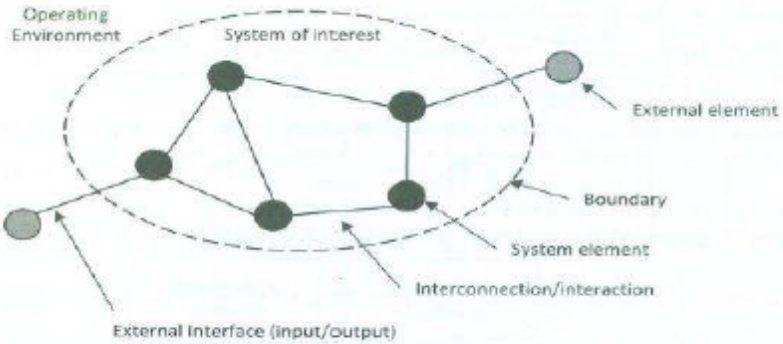


Figure 1-2. An SOI: its elements, interconnections, boundary, and interfaces to external elements in the operating environment.

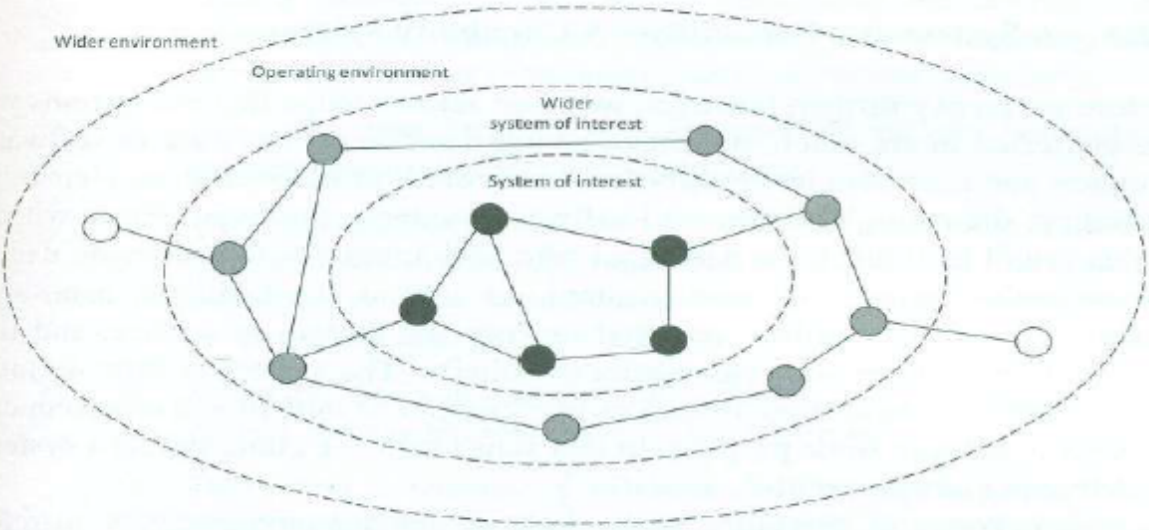
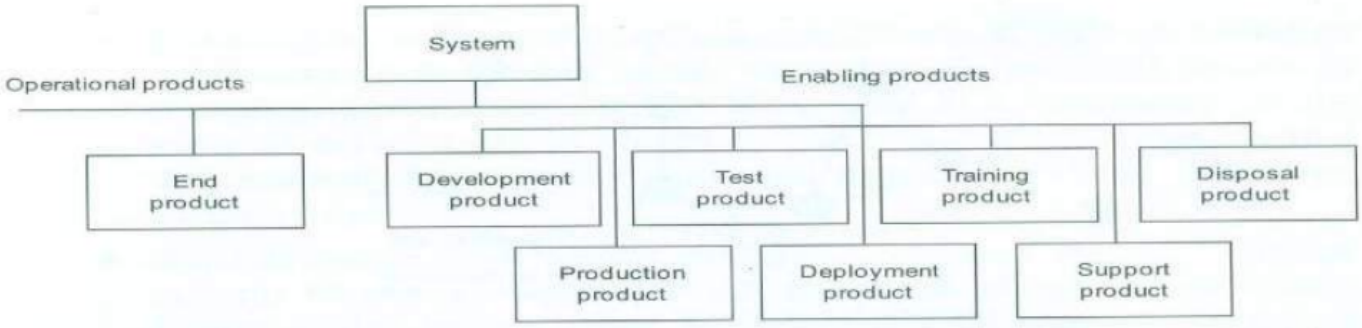


Figure 1-3. An SOI, which might be considered as part of a WSOI within an operating environment, which can be conceived as being part of a wider environment.

System as a Product

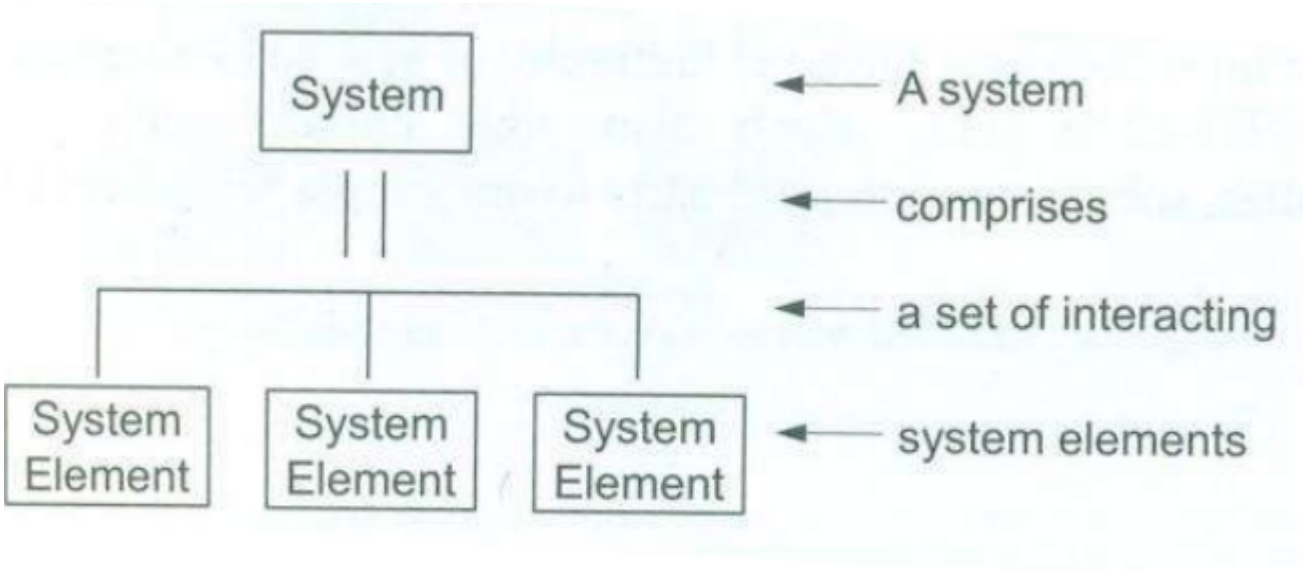


Logical and Physical description of a system

- **Logical description (Functional description)**:
  - What the system will do?
  - How well it will do it?
  - How it will be tested?
  - Under what conditions it will perform?
  - What are the other systems that will be involved with its operation?
- **Physical description**:
  - What are the system elements?
  - How are system elements manufactured, integrated and tested?

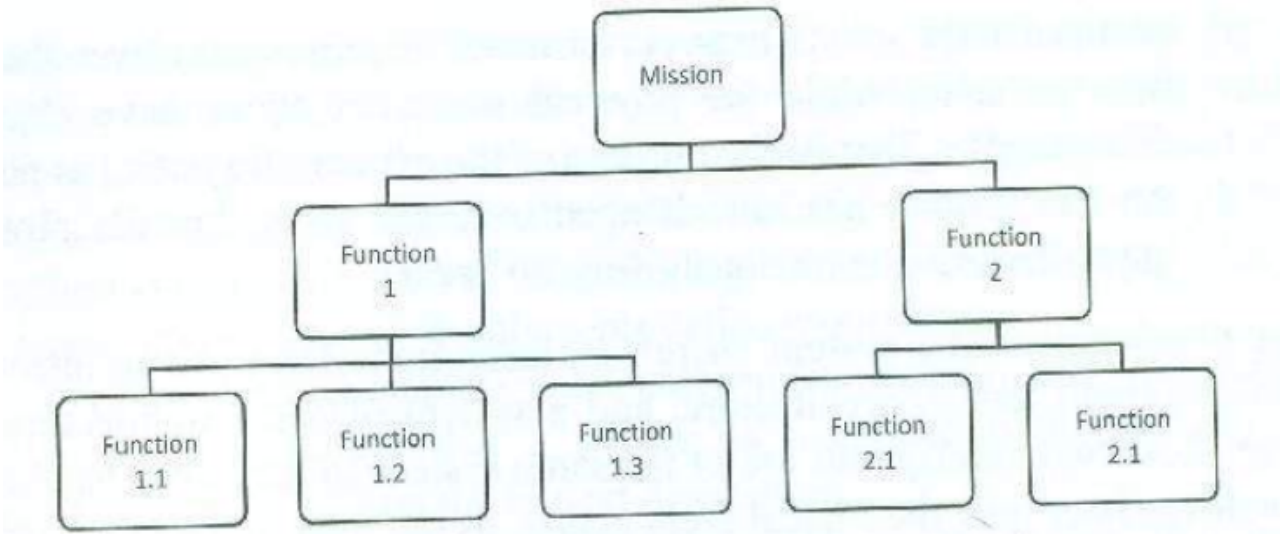
Hirerachical Descriptions of a system

- A system as a combination of system elements which interact to achieve a defined mission.
- Hirerachical description contains allocation of specific functions to system elements to contribute to the system's mission.

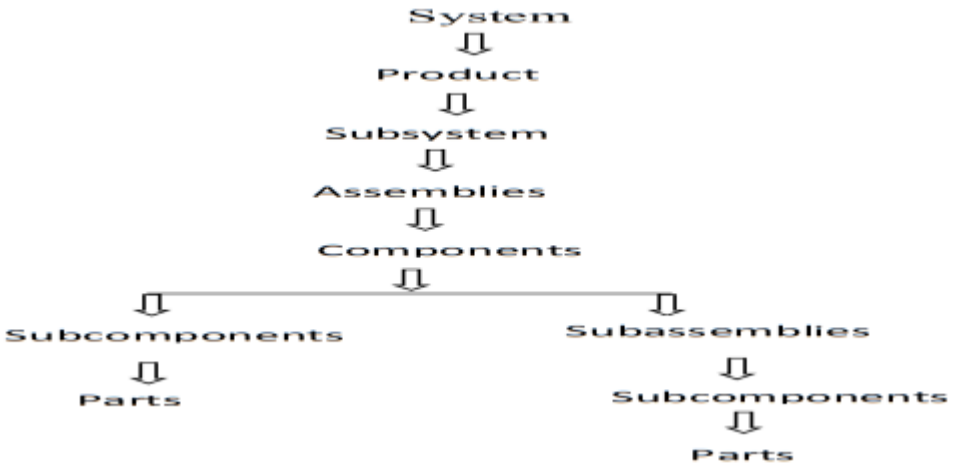


- **Logical Hierarchy**:
  - Description of a system where the system's mission is broken down into a hierarchical structure of its major functions.

- Also known as `functional hierarchy` or `functional architecture`.



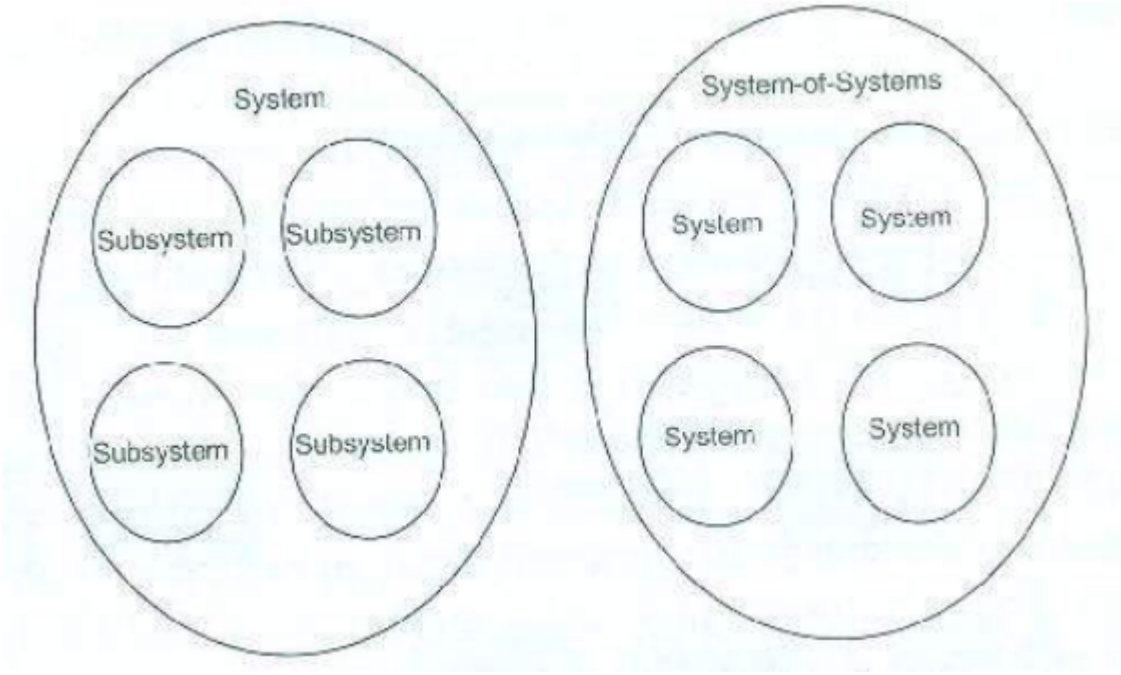
- **Physical Hierarchy**:
  - Description of a system that comprises a number of `subsystems` that comprise a number of `assemblies` that comprise a number of `components`.



System of Systems (SoS)

- Collection of multiple, independent systems in context as part of a larger, more complex system.

Subsystem	SoS elements
Dependent	Both managerially and operationally independent
Optimal	Sub-optimal



System Life Cycle

- Sum of all activities that build on the results of the preceding phase or activity throughout the life of a system.
- Phases:

Phase	Objectives	Key Activities	Outputs
Pre-acquisition	Identify needs, Feasibility analysis and Planning	Requirement analysis, Feasibility studies and Conceptual design	Feasibility Report and System Requirement Specification
Acquisition	Select solution, Design and Development, Testing	Procurement or Development, Detailed design, Testing and validation	Developed System, Test Results and Documentation
Utilization	Deployment, Operation and Maintenance, Optimization	System Deployment, Monitoring and Support, Updates and Upgrades	Operational System and Maintenance Records
Retirement and Replacement	Decommissioning, Data Migration and Evaluation	Decommissioning, Data Archiving and Post Implementation Review	Retired System and Evaluation result

Chapter 2: Systems Engineering

What's system engineering?

- Management function which controls the total system development effort for the purpose of achieving an optimum balance of all system elements.
- Process of transforming an operational need into description of system parameters and integrates those parameters to optimize the overall effectiveness.
- Interdisciplinary collaborative approach to derive, evolve, and verify a life cycle balanced system solution which satisfies customer expectations and meets public acceptability.
- Stages:
  - Requirements Engineering
  - System Architecture
  - System Integration
  - Verification and Validation
  - Risk Management
  - Configuration Management
  - System Optimization

System Engineering Approaches

- **Bottom-up approach**:
  - components are combined into assemblies and then into subsystems from which the system is then constructed.
  - `Parts -> Subcomponents -> Components -> Subsystem -> System`
  - The system is then tested for the desired properties and the design is modified in an iterative manner until the system meets the desired criteria.
  - Valid approach for relatively straightforward problems that are well defined.
  - Not applicable for complicated problems.
- **Top-down approach**:
  - Begins by addressing the system as whole, which facilitates anunderstanding of the system and its environment.
  - `System -> Subsystems -> Components -> Subcomponents -> Parts`
  - Terms system, subsystem, assemblies, and component are relative.
  - Valid for managing complicated systems.

Why focus on life-cylce of system engineering?

- About 60% of errors in system engineering occurs in requiremnts analysis phase therefore project managers should focus on each life-cylce in system engineering that the product itself.

System Optimization and Balance

- Systems engineering performs a very important role in system design in ensuring that there is a balance among the various system components.
- It is essential that optimization and balance are managed at the systems level.
- The `top-down` approach in systems engineering is that system optimization and balance can be achieved as a by-product of the design process, something that cannot be guaranteed in a bottom-up design method.

Integration of Disciplines and Specializations

- There are many disciplines (both technical and non-technical) related to systems engineering.
- Examples include `project management` , `logistics management` , `quality assurance` , `requirements engineering` , `software engineering` , `hardware engineering` , and `interface engineering` (or integration engineering).

Management in system engineering

- `Project management` is responsible for ensuring that the system is delivered on-time and within- budget, and meets the expectations of customers.

Benefits of system engineering

- Reduction of the overall schedule.
- Cost reduction during all phases of the system life-cycle.
- Ensures that the user requirements are accurately reflected in the design of the system.
- Reduction of technical risks associated with the product development.

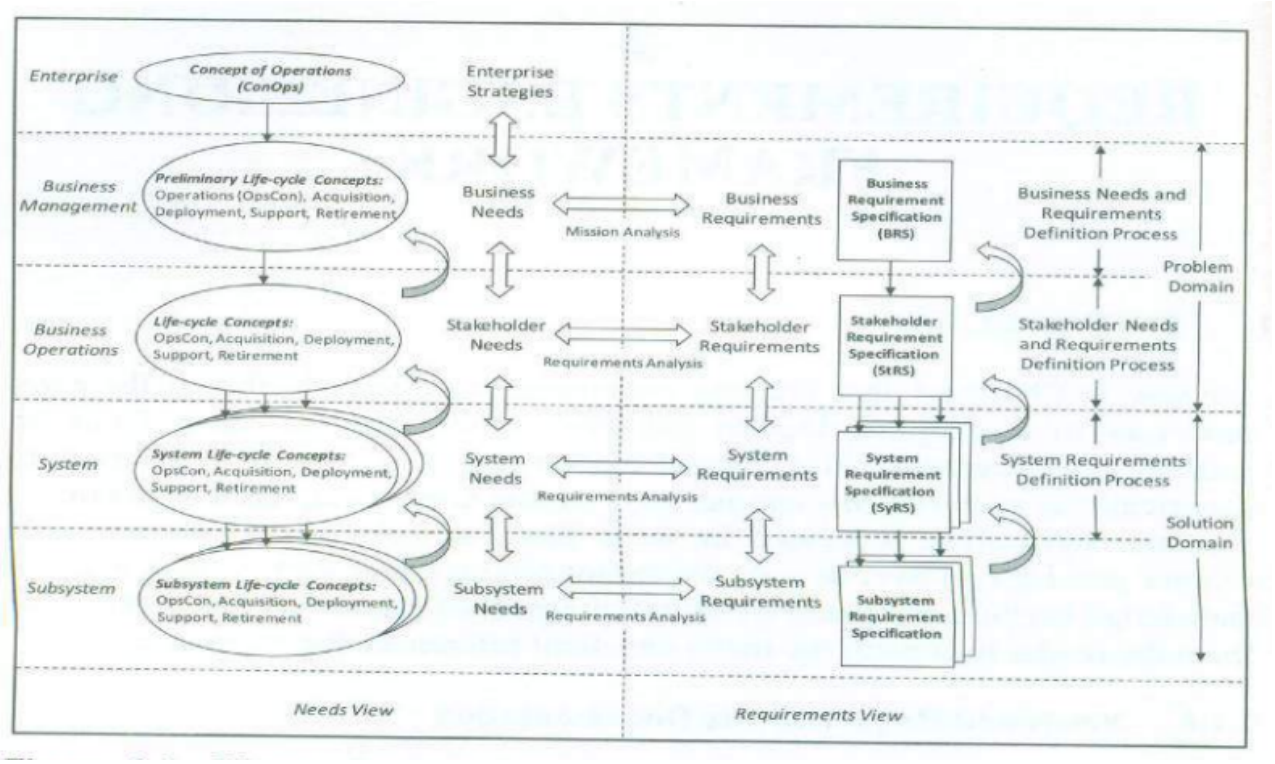
Analysis, Synthesis, and Evaluation

- All extant systems engineering standards and practices extol processes that are built around an iterative application of analysis, synthesis, and evaluation.
- Initially the process is applied at the systems level; it is then re- applied at the subsystem level, and then the assembly level, and so on until the entire development process is complete.
- During the earlier stages the customer is heavily involved; in the latter stages, the contractor is mainly responsible for the continuing effort, which is monitored by the customer.
- `Analysis -> Synthesis -> Evaluation` .

Chapter 3: Requirement Engineering

Need and Requirements documentation

- Needs and requirements are used to capture and document the desired outcomes and specifications for a system.
- Need and requirements documentation is process of recording the needs and requirements of stackholders to ensure a clear understandingof what the system should achieve and how it should function.
- Steps:
    - Identify Stackholders
    - Elicitation of Needs
    - Categorize and Prioritize Needs
    - Requirements Definition
    - Requirements Documentation
    - Validation and Verification
- Needs and Requirements at system levels:
    - Problem domain:
        - **Enterprise view**: enterprise leadership sets the enterprise strategies and concepts of operations.
        - **Business management view**: business management derive business needs and constraints as well a formalize their requirements.
        - **Business operation view**: stakeholders define their needs and requirements.
    - Solution domain:
        - **System view**: the system is defined in logical and physical views.



Need vs Requirement

- **Needs** are capabilities stated in the language of the business at the business management or business operations levels.
- **Requirements** are formal structured statements that can be validated-there may be more than one requirement defined for any one need.

What's Requirement?

- Statement of a system service, an attribute or a quality of the system, or a constraint placed on the system.
- Something the system must do.
- A quality or attribute that it must possess.
- A constriant under which it must operate or be developed.

Requirements Characteristics and Attributes

- **Necessary**: the system couldn't function in desired way without this requirement. If requirement can't be traced by to higer level requirement or associated need, its not necessary.
- **Singular**: requirement can't be combination of two or more requirements.
- **Correct**: requirement must be correct to meet the real needs.
- **Unambigous**: All readers of the requirement should come to the same understanding of what the requirement is saying. There can be only one interpretation.

- **Feasible**: requirement must be achievable using extant design, engineering process andtechnologies.
- **Implementation Independent**: Each requirement must be independent of the method of implementation (process) or physical solution.
- **Justifiable**: Every requirement must be traceable with respect to at least one higher-level requirement.
- **Verifiable**: each requirement must be verifiable.

Requirement Attributes

- To support analysis and management, requirements should be recorded with associated information including the following attributes `Unique identifier`, `Short title`, `Priority`, `Criticality`, `Feasibility`, `Risk`, `Source`, `Type`, `Rationale` and `Relationship to other requirements`.

Emergent Properties

- Are those that are possessed by the system as a whole.
- Only emerge after all the individual subsystems have been integrated.
- Cannot be exhibited by parts of the system in isolation.
- Depend on interactions between components, including their environment.
- Must be defined in the top-down approach.
- Example: maintainability, rehabilitee, availability, usability, security, safety, comfort and ease of use.

What's requirement engineering?

- The process by which we identify a problem's context, locate the business and stakeholder needs and requirements within that context, and deliver specifications that meets those needs.
- Science and discipline concerned with analysing and documenting requirements.(Dorfman and Thayer)
- The systematic process of eliciting, understanding, analysing, documenting, and managing requirements.(Kotonya and Sommerville)
- Process through which the acquirer and the suppliers of a system discover, review, articulate, understand, and document the requirements on the system and the life cycle processes.(ISO)

Why do we need requirements in system engineering?

- To understand stackholder's needs
- To use as guiding roadmap for development process
- To define the scope of the system
- To set quality standards
- To manage complexity

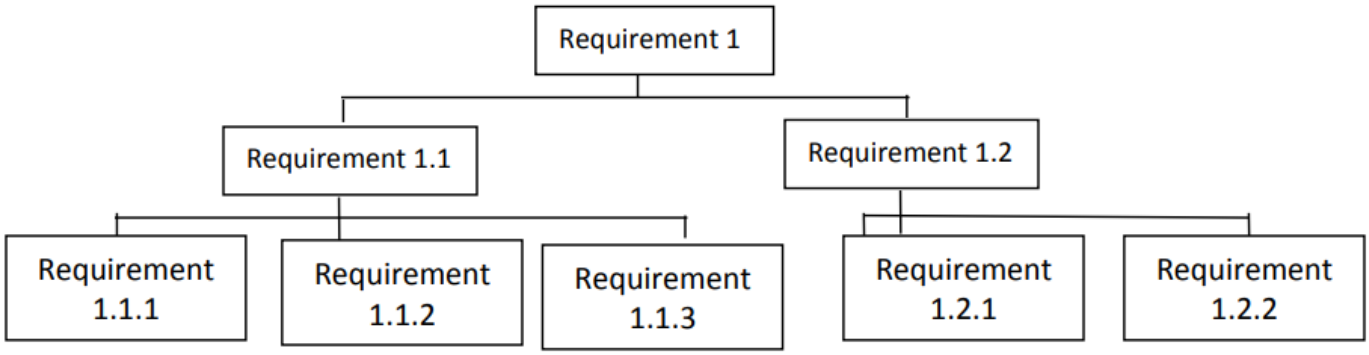
- To facilitate communication and collabration
- To manage change
- To ensure customer satisfaction

Why do we need requirements engineering?

- To understand stackholder's needs
- To minimize risks and costs
- To enhance communication and collabration
- To improve solution quality
- To manage scope and change
- To traceability and accountability
- To support decision making
- To enable validation and verification

Requirements Elicitation and Elaboration

- **Requirements Elicitation**:
    - process of gathering information from stackholders to identify their needs, expectations and requirements for a system, software or a product.
    - techniques:
        - Interview: 1-1 discussions with stackholders to gain insights into their needs and expectations.
        - Workshop: collabration session involving multiple stakeholders to gather requirements collectively.
        - Survey: questionaries or forms distributed to stackholders to gather their input.
        - Observation: directly observing users or system interactions to understand requirements.
        - Prototyping: creating mockups or prototypes to gather feedback and refine requirements.
- **Requirements Elaboration**:
    - process of refining and expanding the initially captured requirements to provide more detail and clarity.
    - activities:
        - Requirement Analysis
        - Requirement Prioritization
        - Requirement Specification
        - Requirement Validation



Difficulties in Requirements Elicitation and Elaboration

- Incomplete or Inconsistent Stakeholder Input
- Ambiguity and Lack of Clarity
- Changing Requirements
- Conflicting Requirements
- Domain Complexity and Technical Expertise
- Communication and Collaboration Issues
- Unrealistic or Infeasible Expectations
- Lack of User Involvement

Requirement Validation

- Process of evaluating and verifying the captured requirements to ensure that they are complete, consistent, correct, and aligned with stakeholder expectations.
- An essential part of the requirements engineering process and helps in identifying and addressing any issues or problems with the requirements before proceeding with system development.
- Activities:
  - Review and Inspection
  - Prototyping and Simulation
  - Requirements Traceability
  - Stakeholder and User Feedback and Confirmation
  - Validation Testing
  - Requirement Metrics and Quality Criteria

Requirement Documentation

- Writing well-formed requirements.
- Guidelines:
  - Ensure each requirement is a necessary, short, definitive statement.
  - Define the appropriate conditions (qualitative or quantitative measures) for each requirement.

    - Ensure that each requirement is verifiable associated with writing a verification requirement statement.
    - Avoid over-specification, unnecessary constraints and unbounded statements.
    - Ensure readability by defining standard templates for describing requirements using natural language.

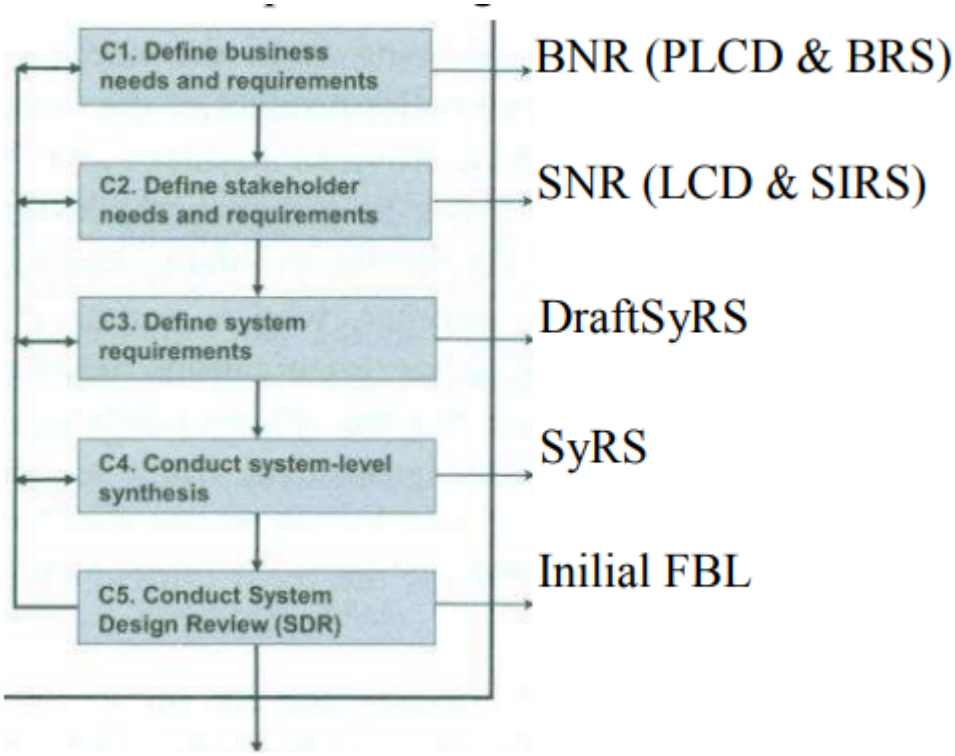
Requirements Management

- process of effectively managing and controlling the requirements throughout the entire lifecycle of a system, software or a product.
- activities aimed at capturing, organizing, prioritizing, tracing, and maintaining requirements to ensure their completeness, consistency, and alignment with stakeholder needs and project objectives.
- key aspects:
    - Requirement Change Management
    - Change Management Traceability:
        - Froward traceability
        - Backward traceability
    - Requirement management tools:
        - software applications designed to assist in the effective management, documentation, analysis, and traceability of requirements throughout the project lifecycle.
        - features:
            - Supports requirements elicitation
            - Allows readers to browse
            - Allows specific requirements to be retrieved
            - Supports forward and backward traceability
            - Supports the generation of good requirements
            - Generates reports
            - Allows import and export of requirements in various formats
            - Supports change control and change impact assessment.
            - Supports allocation and logical-to-physical translation
            - Does not enforce a particular requirements engineering process
        - Example: `IBM Rational DOORS`, `Jama Connect`, `Microsoft Azure DevOps`, `Atlassian Jira` and `Gatherspace`.

Chapter 4: Conceptual Design

Conceptual Design

- First activity in Acquisition Phase.
- Aims to articulate the needs, to analyse and document the system-level requirements flowing from the needs, and to complete a logical design of the system.
- Responsible for the expansion of the system definition from relatively brief business needs into a logical set of system-level requirements that may be hundreds of pages long.
- Concerned with the transition from the problem domain into the solution domain.
- **Functional Baseline**:
  - system-level logical architecture that is the basis for subsequent lower-level (physical) design.
  - represents a system-level description of the functions and capabilities that the system should provide.

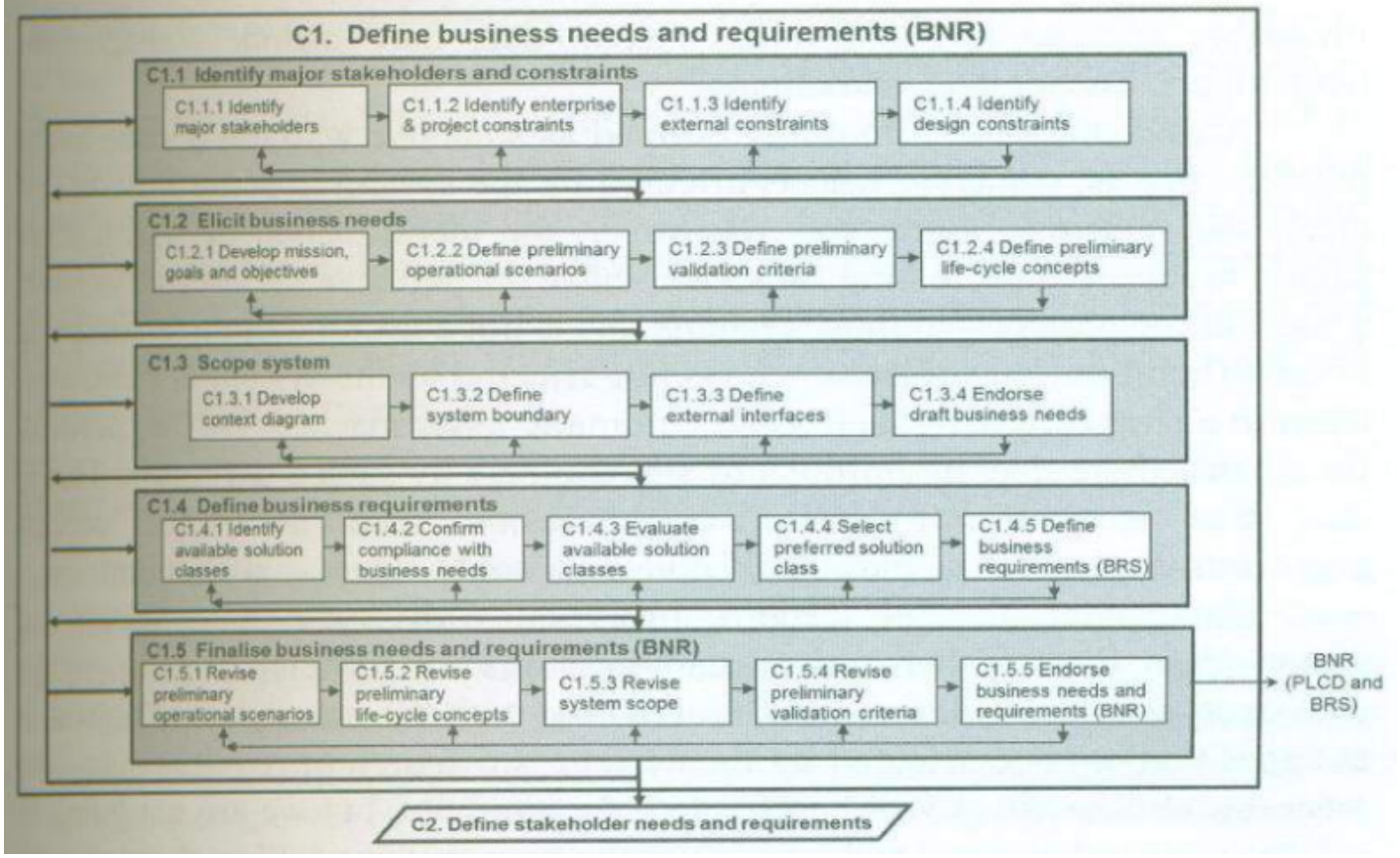


Define Business Needs and Requirements

- Identify Major Stakeholders and Constraints:
  - A `stakeholder` is commonly defined as someone who has a stake in the project-that is, someone who is affected by the system in some way, or can affect the system in some way.
  - Stages:
    - Identify Major Stakeholders
    - Identify Business and Project Constraints

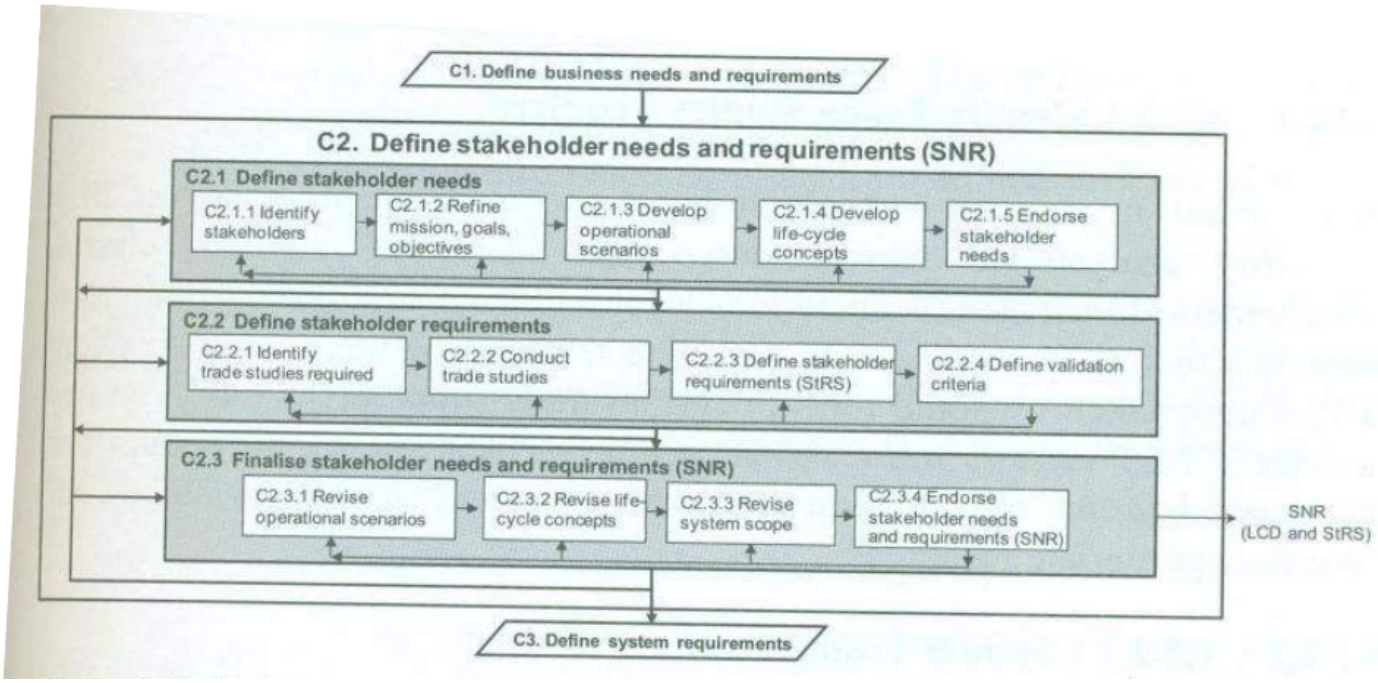
- Identify External Constraints - Identify Design Constraints

- Ellicit Business needs:
  - Define Mission, Goals, and Objectives:
    - Guidelines for writing missions:
      - A single sentence
      - No conjunctions
      - Contain no more than 5-7 concepts
      - Include an ‘in order to’ clause
      - Avoid physical terms
      - Rely on iteration
  - Define Preliminary Operational Scenarios
  - Define Preliminary Validation Criteria:
    - any mechanism by which the customer will measure satisfaction with the products of the Acquisition Phase.
    - Key criteria include `performance` in each operational scenario, `safety`, `reliability`, `supportability`, `maintainability`, `ease of use`, and `time and cost to train`.
    - `Measurability`, `Baseline`, and `Relevance` needs to be considered when identifying validation criteria.
  - Define Preliminary Life-cycle Concepts
- Scope Systems:
  - Develop Context Diagram
  - Define System Boundary
  - Define External Interfaces:
    - External interfaces are between the system of interest and each of the other existing or future external systems to which it is interconnected.
  - Endorse Draft Business Needs
- Define Business Requirements:
  - Feasibility Analysis:
    - Identify the alternative system-level solution classes capable of satisfying the Business Needs.
    - Confirm compliance of each solution class with the Business Needs and, if not completely compliant, note the likely level of achievement.
    - Evaluate the alternative solution classes in terms of feasibility, performance, effectiveness, technical and project risk, and other selected measures.
    - Select the best of the alternative solution classes (ensuring that the options are narrowed down as much as is practicable at this stage).
  - Define Business Requirements (BRS)
- Finalise Business Needs and Requirements (BNR)



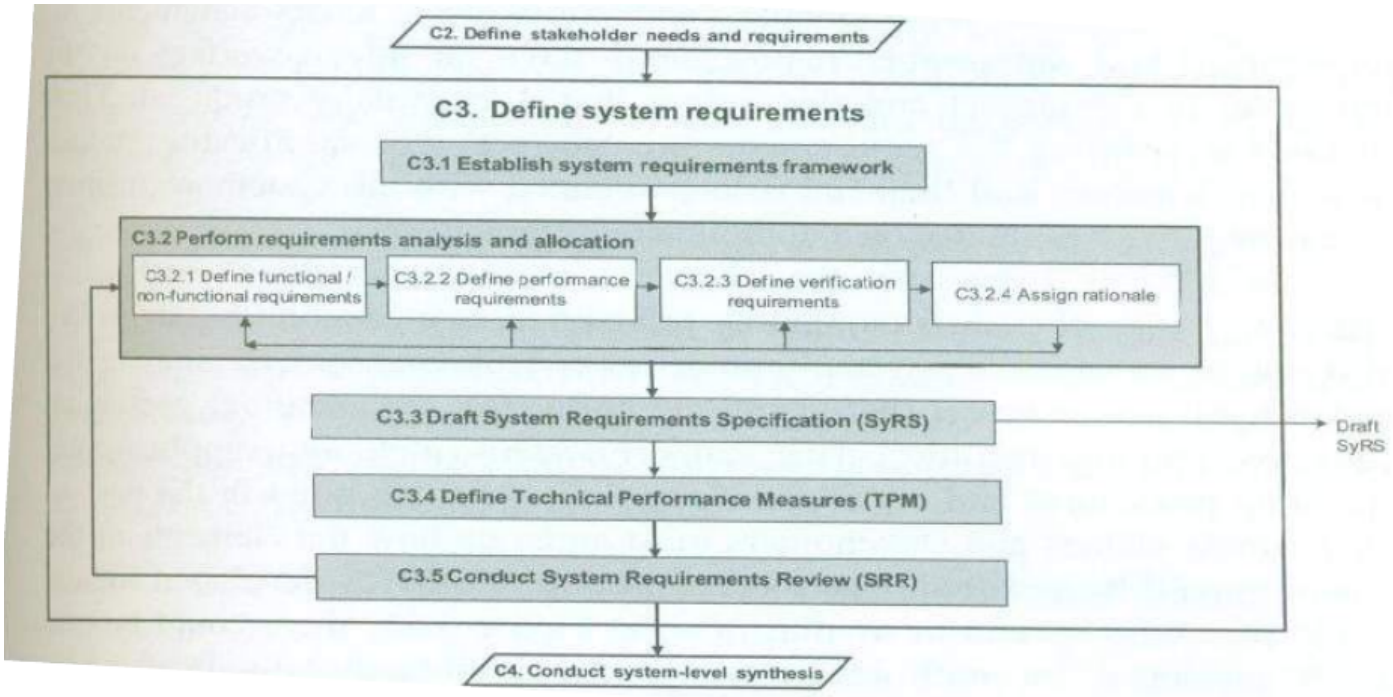
Define Stakeholder Needs and Requirements

- Define Stakeholder Needs:
  - Identify Stakeholders
  - Refine Mission, Goal, Objectives
  - Develop Operational Scenarios
  - Develop Life Cycle Concepts
  - Endorse Stakeholders Needs
- Define Stakeholders Requirements:
  - Identify Trade Studies Required
  - Conduct Trade Studies
  - Define Stakeholder Requirements (StRS)
  - Define Validation Criteria
- Finalise Stakeholder Needs and Requirements (SR):
  - Revise operational scenarios
  - Revise life cycle concepts
  - Revise system scope
  - Endorse stakeholders needs and requirements (SNR)



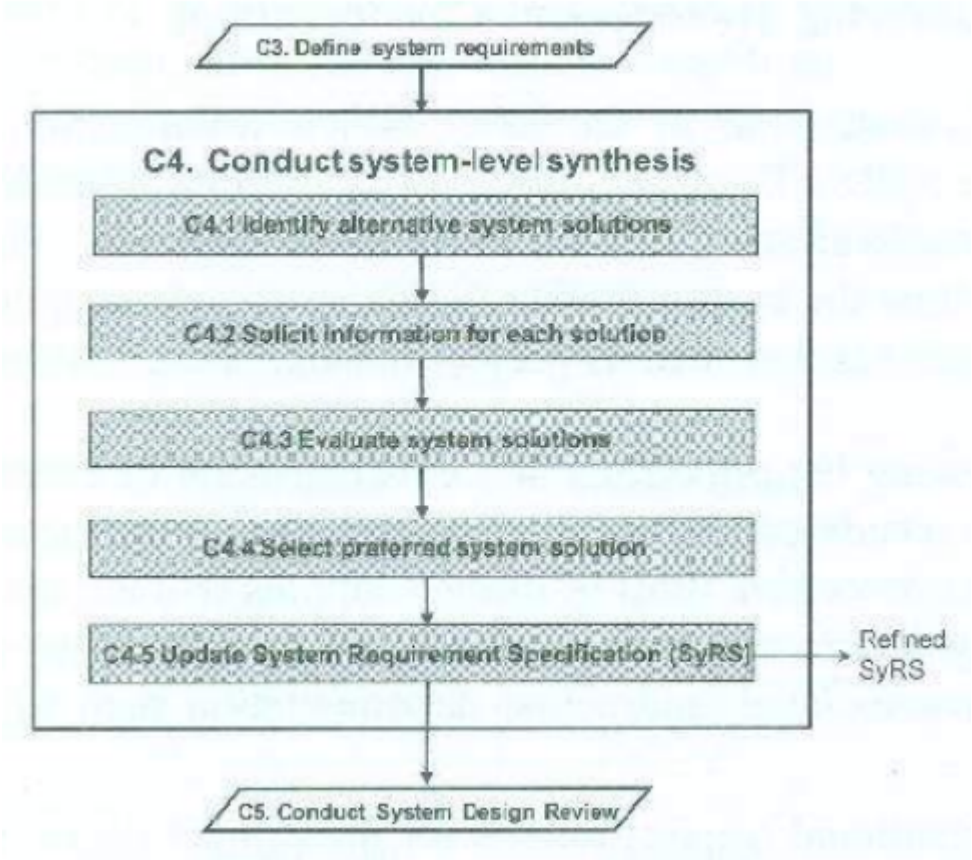
Define System Requirements

- Establish Requirements Framework
- Perform Requirements Analysis and Allocation:
  - Define Functional/Non-functional Requirements
  - Define Performance Requirements
  - Define Verification Requirements
  - Assign Rationale
- Draft System Requirement Specification (SyRS)
- Define Technical Performance Measures (TPM)
- Conduct System Requirements Reviews (SRR)



Conduct System–Level Synthesis

- Identify alternative system solutions
- Collect information foe each solution
- Evaluate system solutions
- Select preferred system solution
- Update System Requirement Specification (SyRS)



Conduct System Design Review (SDR)

- At the end of Conceptual Design the System Design Review (SDR) provide the following from a systems engineering perspective:
  - formal confirmation that the logical design meets the business and stakeholder requirements.
  - a formal record of design decisions and acceptance.
  - a formalized communication of the intended design approach to the major players in the design effort.
  - approval of the Systems Engineering Management Plan (SEMP) and supporting plans.
- In addition to review of the systems engineering effort, a number of review activities will most likely be performed at SDR to support project management:
  - confirmation that the system to be procured aligns with the customer’s organizational goals.
  - the Project Management Plan (PMP) is refined.
  - cost estimates are refined.
  - the schedule is refined and is confirmed to be consistent with the cost and risk goals for the project.

- confirmation that all required project resources are available.

- Steps:
  - Prepare and review documentation
  - Prepare agenda
  - Confirm entry criteria met
  - Confirm system-level solution
  - Agree action items
  - Confirm exit criteria met

Chapter 5: Preliminary and Detailed Design

Preliminary Design

- Starts with the Initial FBL and continues to translate system-level requirements into design requirements for the system elements that will combine to form the system.

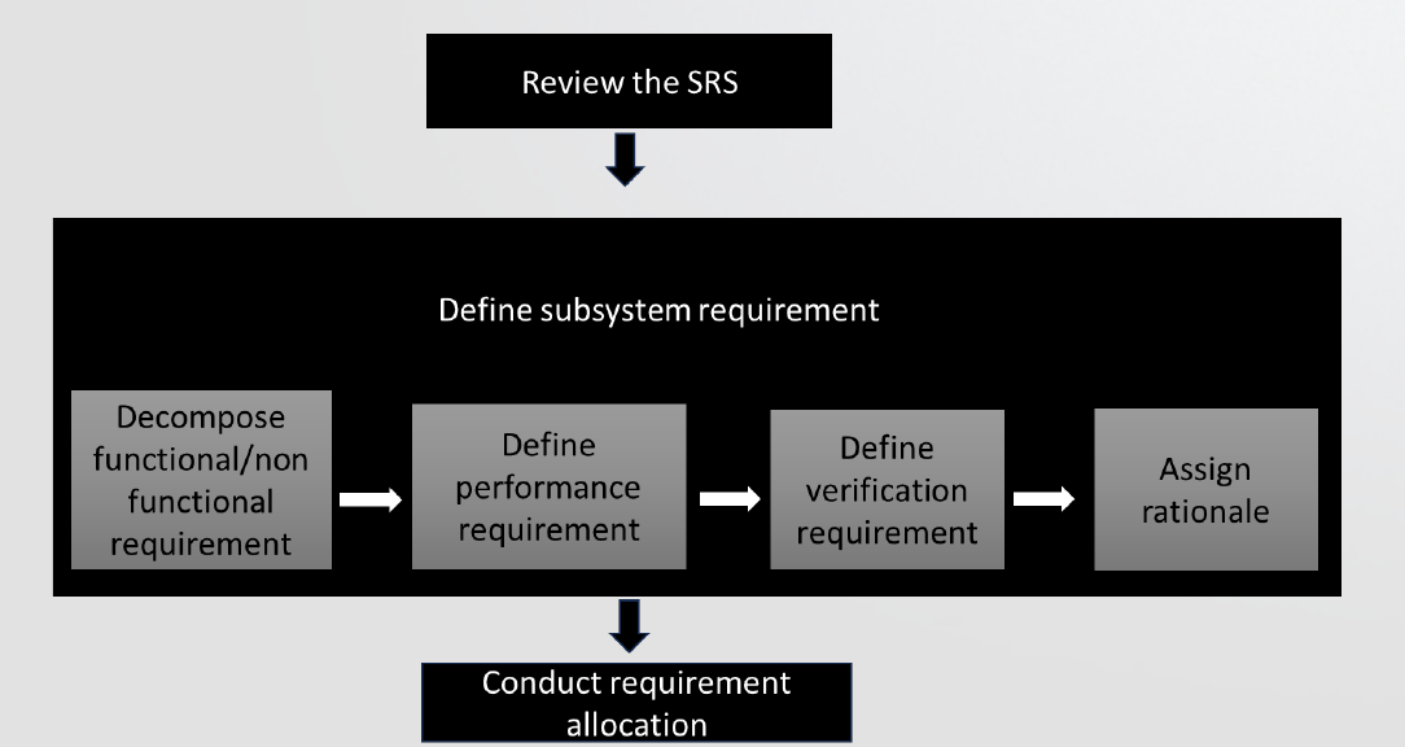
- Trade-off studies are conducted and the result of the Preliminary Design effort is the establishment of an Allocated Baseline (ABL), in which requirements are ‘allocated’ to specific physical system elements that combine to form the system.

- **Activities**:

  - Subsystem requirements analysis.
  - Requirements allocation.
  - Interface identification/design.
  - Subsystem-level synthesis.
  - Preliminary Design Review (PDR).

Subsystem Requirements Analysis

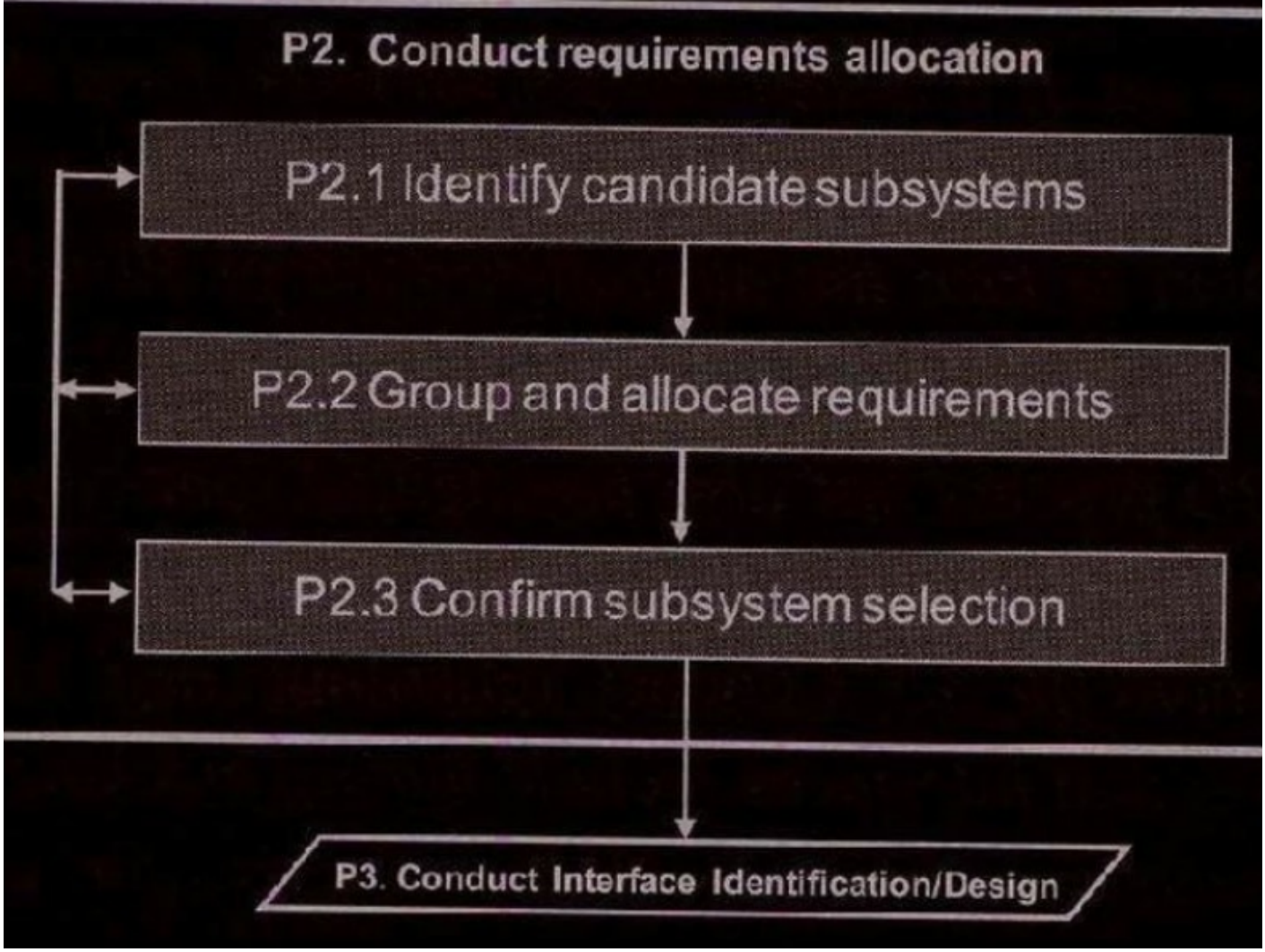
- Elaboration from the system level to the subsystem level and so on until all functions, parameters and interfaces have been defined and the necessary resources have been identified.
- The detail must be sufficient to define completely the functions required of the major subsystems comprising the system, and must be able to be used either to procure, or to design and produce, the major subsystems and their assemblies and components during subsequent stages of the system engineering process.



- The diagrams that are generated during the Preliminary Design effort are more detailed than in Conceptual Design and are aimed at defining the subsystems (rather than the system).
- This stage is therefore the beginning of the conversion of the ‘what’ the system needs to do into the ‘how’ it is going to do it, which is more commonly referred to as the translation of logical design into physical design (or from the problem domain into the solution domain).

Requirements Allocation

- Takes new dimension by forming groups around a preliminary physical architecture crafted by designers.
- The identified groups of similar functions become pivotal in determining the design of major system elements, referred to as `Configuration Items` (CIs).
- These CIs are either selected or designed to execute the group of requirements assigned to them.
- The primary focus during this phase is the allocation of requirements to these Configuration Items.



- **Significance**:
  - It provides a structured approach for designers to define the major elements of the system, ensuring that each Configuration Item is tailored to perform the specific set of requirements assigned to it.
  - It streamlines the design process, facilitating a systematic and organized development of the major system components.

Requirements Breakdown Structure and Work Breakdown Structure

- **Requirements Breakdown Structure (RBS)**:
  - A hierarchical representation that systematically decomposes and organizes the logical requirements of a system.
  - It outlines the relationships and dependencies between various requirements, helping to structure and categorize the essential functionalities and characteristics needed for the successful development of a system.
- **Work Breakdown Structure (WBS)**:
  - A hierarchical decomposition of the total scope of work to be carried out by a project.
  - It breaks down the project into smaller, more manageable components, known as work packages.
  - It's organized in a way that reflects the physical structure of the project and serves as a foundation for project planning, management, and control.

RBS Vs WBS

- The RBS articulates the logical design, representing the breakdown of logical requirements into subsystems or Configuration Items (CIs). On the other hand, the WBS documents the products and work packages needed for system development, encompassing the design, development, integration, andtesting of subsystems/CIs.
- While RBS focuses on logically organizing requirements, the WBS is structured physically, emphasizing project management imperatives.
- The WBS includes not only the subsystems/CIs but also encompasses additional work related to design, development, integration, and testing.

Interface Identification and Design

- Interface are identified during the selection of the subsystems/CIs.
- Identification of interfaces not only determine successful operation of the system once integrated, but they also place additional limitations and requirements on the design of the individual subsystems/CIs.

Importance of Interface Management

- Interface management is crucial in systems engineering due to the prevalence of problems that often come in at the interfaces between different components or subsystems.
- Recognizing and defining interfaces with precision can mitigate risks associated with miscommunications, compatibility issues, and errors during the integration and operation phases of a system.
- Effective interface management contributes to the overall reliability, performance, and longevity of a system.

Types of Interfaces

- Physical interfaces
- Hydraulic/pneumatic interfaces
- Software interfaces
- Environmental interfaces

Interface Induced Failures

- Refer to issues and problems within a system that arise specifically due to inadequacies or problems at the interfaces between different components, subsystems, or systems.
- Neglect and Weak Points
- Bottlenecks
- Structural Failures
- Erroneous Function Calls
- Compatibility Issues
- Mitigation and Prevention

Interface Control Documents

- When two or more CIs have an interface, the teams developing those CIs will no doubt have an interest in and specific requirements for that interface.
- To ensure that the teams developing the CIs have their needs met, and to facilitate communication between the teams, an interface control working group (ICWG) may be formed to bring together those involved in the interface definition and design.
- ICWGs must meet regularly to generate the initial ICD and should then meet periodically or as the need arises to address any ongoing issues. These meetings and reviews may be informal but the products of the meetings are formal interface design issues and decisions, and updated ICDs.

Subsystem-level Synthesis and Evaluation

- Subsystem-Level Synthesis refers to the process within systems engineering where individual subsystems are designed, developed, and synthesized to fulfill specific functions or tasks within a larger system.
- This phase occurs after the overall system requirements have been defined, and it involves breaking down the system into smaller, manageable components known as subsystems.

Investigate Design Alternatives

- The initial synthesis is conducted to determine a preliminary design that identify the specification using one of the three broad design options available.
- **Design options**:
  - Commercial-off-the-shelf (COTS),
  - Modified COTS, and
  - Developmental items.
- Factors that impact on the designer's decision to use COTS, modified COTS or specially developed items include the specific requirements to be performed by the item, the availability and stability of the current technology, size of the market, supportability and cost, and any contractual directives. The following sections discuss each of these options.

Preliminary Design Review (PDR)

- A structured assessment and evaluation of the initial design concepts and plans for a project or system. This review takes place after the completion of the conceptual design phase and before moving into the detailed design phase.
- The primary purpose of the PDR is to examine the proposed design, ensuring that it aligns with the project requirements and objectives.
- It serves as a checkpoint to validate the feasibility and soundness of the design before significant resources are committed to the detailed design and development phases.
- **Key aspects**:
  - Design Concepts
  - Requirements Compliance
  - Risk Assessment

    - Feasibility
    - Cost and Schedule Estimates
- **Outcomes**:
    - Approval to proceed to the next phase of detailed design and development.
    - It may lead to recommendations for design modifications, additional analyses, or further refinement before moving forward.
- **Benefits**:
    - Help in identifying and addressing design issues early in the development process, reducing the likelihood of costly changes later.
    - It ensures that the design aligns with the project goals, meets requirements, and is technically and economically feasible.
- **Tools**:
    - Schematic Block Diagrams
    - Physical modellling
    - Mathematical modelling and simulation
    - Trade-off analysis

Detailed Design and Development

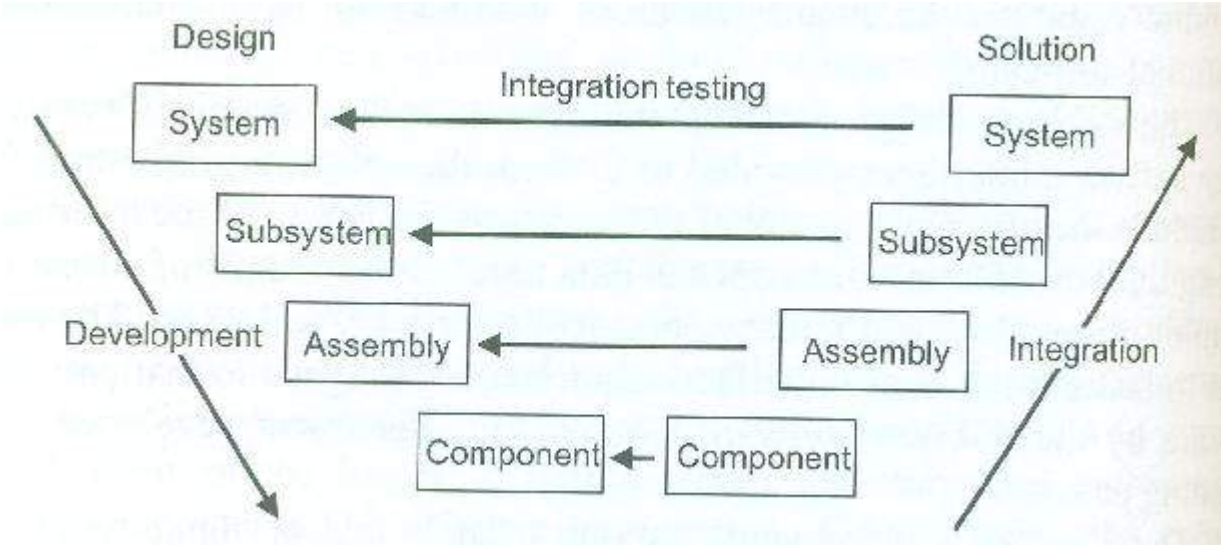
- Continues the development effort and makes use of the FBL and ABL developed during Conceptual Design and Preliminary Design.
- The detailed design effort takes these definitions of the overall system and of the major CIs and finalizes the design of specific components that make up the CIs.
- The realization and documentation of individual components used to support production is referred to as the `Product Baseline` (PBL).
- **Activities**:
    - Describing the lower-level assemblies and components making up the CIs (and their interrelationships).
    - Defining the characteristics of the above items through specifications and design data.
    - Finalizing the design of all interfaces necessary to support system integration.
    - Either procuring the above items off-the-shelf, or designing them if they are unique to the system under development.
    - Developing prototypes or engineering models of the CIs by integrating the above items (for the purpose of design verification, and system-level test and evaluation).
    - Redesigning and retesting (as required).
    - Conducting the Critical Design Review (CDR) to confirm that the design is ready for construction and production.

Detailed Software Design Processes

- Defining the scope
- Preparing for Software Design
- Performing Detailed Software Design
- Documenting the design

Integrating System Elements

- At this stage, All CIs (hardware and software) and their interfaces are designed and tested.
- Low-level items are integrated into higher-level assemblies and tested against requirements.
- COTS items are procured and checked against supplier specifications.
- System-level prototypes are created to confirm design requirements.
- This prototype can be used to confirm that the system-level design requirements have been met.



Detailed Design Reviews

- Conducted at the conclusion of Detailed Design and Development to ensure that the system design is complete prior to Construction and/or Production.
- **Equipment/Software Design Reviews**:
  - Also known as Software Design Reviews.
  - Focused on particular items of software and hardware to ensure that the specific design approach meets the requirements.
  - Associated documentation such as trade-off reports that justify and explain the design approach will also normally be investigated during these reviews.
  - These reviews investigate the drawings, computer programs, material lists and the necessary specifications to thoroughly review the item under investigation.
  - All equipment/software design reviews will be completed prior to the major formal review of the phase, which is called the `Critical Design Review` (CDR).
- **Critical Design Review (CDR)**:
  - The final review before construction/production.
  - CDR aims to satisfy requirements, confirm compatibility, and finalize test/production plans.
  - CDR establishes the Product Baseline and freezes design activities.
  - Aims to:
    - evaluate the detailed design which is documented in various forms including the mature Product Specifications and related engineering drawings.
    - determine readiness for production/construction.
    - determine maturity of software.
    - determine design compatibility.

        - establish the Product Baseline (PBL)

Chapter 6: Construction, Production and Utilization

Definition

- **Construction**:
    - Involves the physical creation of buildings, infrastructure, and other structures.
    - It includes activities such as site preparation, foundation laying, framing, roofing, and finishing.
    - Requires careful planning, skilled labor, and the user of various materials and equipments.
- **Production**:
    - Refers to the manufacturing or assembly of goods or products.
    - This process involves converting raw materials or components into finished products through various manufacturing techniques such as machining, molding, welding, and assembly.
    - Requires specialized machinery and skilled workers to ensure efficient and high-quality output.
- **Utilization**:
    - Involves the deployment or use of products or structures for their intended purpose.
    - The effective utilization of products often requires proper maintenance and management to ensure longevity and functionality.

Objectives

- Meeting customer requirements.
- Optimizing resource utilization.
- Ensuring quality control
- Managing risks.
- Adhering to schedules.
- Enhancing efficiency.

Production Requirements

- Production requirements need to be considered early in the Acquisition Phase to ensure that production risks are identified and addressed as early as possible.
- As with all other technical requirements, the earlier that production issues are identified in the acquisition, the easier they can be addressed.
- Typical construction and production issues that need to be addressed and monitored throughout the entire systems engineering process include:
    - Material availability (lead times), ordering, and handling.
    - Availability of skill sets (including any training).
    - Availability of production tools and equipment.

    - Fabrication requirements including production requirement, assembly drawings and instructions.
    - Processing and process control.
    - Assembly, inspection and test and.
    - Packaging, storage, and handling.

    - Production plan contains:
        - Resources
        - Production engineering considerations (Schedules and manufacturing methods and processes)
        - Materials and purchased parts
        - Management
        - Logistics

Engineering Management Issues

- Before the system completes Construction and/or Production, it should be subjected to a range of acceptance tests.
- Acceptance testing, alone, is insufficient. Audits should be conducted during production and/or construction to ensure that the system has been built in accordance with the approved specifications and project documentation.

Configuration Audits

- **Functional configuration audit**: used to verify and certify that the performance of the CI meets the specified requirements.
- **Physical configuration audit**: provide confidence that the as-built CIs match the low-level specifications such as the Product Specifications, assembly specifications, drawings and technical data.

Major System Reviews

- **Test Readiness Review (TRR)**:

    - Demonstrate that the system CIs are ready to enter formal test and evaluation.

    - Contains:

        - Test and Evaluation Master Plan (TEMP)
        - Test plans
        - Formal and informal test results
        - Supporting documentation
        - Support, test equipment, and facilities

- **Formal Qualification Review (FQR)**:
    - Verify that the performance of each of the CIs meets all the functional requirements when integrated together into the system.
    - Demonstrate that the integrated system complies with all the specifications generated as part of the project including SyRS , hardware and software Development Specifications, and interface requirement specifications.
    - Considered as `system-level` configuration audit.

Major Activities During Utilization Phase

- `Operational use`
- `System support`：
    - Ensure that the system is maintained in operational order.
    - The system support will be conducted in accordance with the approved logistics concept.
- `Modifications`：
    - Required to ensure that the system continues to meet operational and support requirements.
    - Modifications made in accordance with configuration management practices enhance supportability and operations, but unmanaged modifications adversely impact training, support, safety and operations in the long run.
    - Uses:
        - To rectify discrepancies with the performance of the system that were not identified during the Acquisition Phase.
        - To changing system-level requirements.
        - To increase the efficiency of the system, or perhaps reduce maintenance costs, through the replacement of system elements.
    - The degree of impact depends on the size and significance of the modification.

Failure Reporting, Analysis and Corrective Action System (FRACAS)

- A closed-loop system designed to continually maintain visibility into system operation and support.
- **Steps**:
    i. Failure Reporting: accurately documenting and reporting all failures and incidents that occur within the system.
    ii. Data Collection: gathering comprehensive data on failure occurrences, including their nature, frequency, and impact on the system.
    iii. Failure Analysis: conducting thorough investigations to identify the root causes of failures using systematic analytical methods.
    iv. Corrective Action Planning: developing and planning corrective actions based on the findings from the failure analysis phase.
    v. Corrective Action Implementation: executing the planned corrective actions to address the identified root causes and prevent recurrence of similar failures.

vi. Monitoring and Feedback: monitoring the effectiveness of implemented corrective actions and incorporating feedback into the FRACAS process to drive continuous improvement.

Retirement

- The final stage of a system's life cycle, involving phase-out and disposal functions.
- It's crucial to develop a Retirement Concept during the early stage of Acquisition Phase, considering disposal and phase-out issues.
- Designing for retirement, also known as "design for disposability," involves considering disposal and phase-out issues as criteria for system design.

Steps in Retirement Design

1. **Identify the reasons for potential retirement**:
    - The system may have reached the end of its usable and supportable life.
    - Critical components of the system may be damaged or no longer supportable.
    - The system may still be useful, but the business need for it has disappeared or the business owner is forced to retire it due to various issues.
    - The system or its critical components may be damaged beyond repair.
2. **Identify potential retirement methods for the system**:
    - Sold as a second-hand item.
    - Traded in on a replacement system.
    - Re-deployed as a training aid.
    - Destroyed and disposed as waste Retirement.
    - Disassembled-that is, broken up and sold as working sub-systems.
    - Scrapped-that is, broken up and sold as sub-systems, assemblies.
    - Placed in storage-that is, mothballed because none of the other methods are currently acceptable or tenable.
3. **Identify design issues that arise**:
    - observance of recycling regulations.
    - avoidance of the use of potentially hazardous/toxic materials.
    - environmental impact.
    - cost of disposal/destruction/uninstallation.
    - cost of refurbishment for resale or trade-in.
    - desirability of salvage or material recovery for resale.
    - cost of transportation for disposal.
    - need for specialized personnel/equipment required for disposal

Chapter 7: System Engineering Management

Definition

- Systems Engineering Management (SEM) is a discipline that focuses on the management of complex engineering systems throughout their lifecycle.

Benefits of Applying SEM to Software Development

- Improved planning and control
- Enhanced system quality and performance
- Reduced risk and development costs
- Effective stakeholder communication and collabration

Key SEM practices for Software Development

- Requirement Engineering
- System Architecture
- Integration and Testing
- Configuration Management
- Risk Management
- Project Management

Technical Review and Audit Management

- **Types of reviews**:
    - Design reviews: evaluate system design to meet functional and performance requirements.
    - Code reviews: asses source code for errors, adherence to coding standards, and maintainability.
    - Documentation reviews: ensure completeness, clarity, and accuracy of project documentation.
    - Configuration Management reviews: verify control, tracking, and management of software versions and configurations.
- **Review Process**:
    - Planning: define objectives, scope, criteria, and review team.
    - Preparation: collect relevant documentation, specifications, and standards.
    - Execution: conduct review, identify issues, and document findings.
    - Reporting: prepare a report summarizing results, recommendations, and action items.

Benefits of Technical Review and Audit Management

- Improved quality: Early detection and correction of issues leads to higher quality software.
- Risk mitigation: Proactive identification and mitigation of risks reduces potential problems.

- Compliance: Ensures adherence to standards, regulations, and contractual obligations.
- Process improvement: Feedback from reviews guides process improvement for future projects.

Test and Evaluation

- **Testing**:
    - Verifying and validating software functionality, reliability, and performance.
    - Functional testing, Performance testing, Security testing and User acceptance testing.
    - Activities:
        - Test planning:
            - `Test objectives` : clearly define goals and areas to be evaluated.
            - `Test requirements` : identify necessary resources, equipment, facilities, and personnel.
            - `Test strategy` : determine overall approach, types of tests, methodologies, and environments.
            - `Test documentation` : develop detailed test plans, cases, and scripts to guide execution and record expected results.
        - Test case design
        - Test execution: involves the preparation, execution of test cases, data analysis, and defect tracking.
        - Defect tracking
        - Test reporting:
            - `Test results compilation` : compiling and documenting the test results including observed behaviors, performance metrics, identified issues, and deviations from requirements.
            - `Test evaluation` : analysis of test results to determine the overall performance, strengths, weakness, and areas of improvement.
            - `Recommendations` : providing recommendations for addressing identified issues, improving system performance, and mitigating risks.
- **Evaluation**:
    - Performance Assessment: evaluate the system's performance against established metrics and requirements, such as response time, throughput, scalability, and resource utilization.
    - Functional Verification: verify that the system functions as intended and meets the specified functional requirements.
    - Compliance Assessment: assess the system's compliance with relevant standards, regulations, and industry best practices.
    - Risk Identification: identify and evaluate potential risks associated with the system's operation, including safety hazards, security vulnerabilities, or performance limitations.

Risk Management

- **Identifying Risks**: identifying potential risks such as software bugs, security vulnerabilities, or performance issues is a critical aspect of risk management.
- **Implementing Mitigaion**: implementing mitigation strategies to minimize the impact of identified risks and ensure system stability.
- **Continous Monitoring**: continuously monitoring and controlling risks throughout the software development lifecycle is essential for risk management.

Technical Risk Management

- **Risk Identification**: identifying potential technical risks by analyzing the system's design, architecture, interfaces, technologies, and dependencies.
- **Risk Analysis**: assessing the identified risks in terms of their likelihood of occurrence, potential impact, and level of uncertainty.
- **Risk Mitigation Strategies**: developing and implementing proactive strategies to manage and mitigate identified technical risks effectively.

Communication and Documentation

- **Effective Communication**: maintain clear and effective communication channels among project stakeholders regarding technical risks, their status, and mitigation efforts.
- **Risk Documentation**: document the identified risks, risk assessment results, mitigation strategies, and their implementation status.

Configuration Management

- The process of managing and controlling software configurations throughout their lifecycle.
- **Key Principles of Configuration Management**:
  - Configuration Identification
  - Configuration Control
  - Configuration Status Accounting
  - Configuration Audits and Reviews
  - Configuration Management Planning

Specifications and Standards

- **Specifications**:
  - Define the detailed requirements, functionalities, and characteristics of a software product or system.
  - Provide a clear and unambiguous description of the desired features, functionalities, and behavior of a software system.
  - They define the functional and non-functional requirements, user interactions, system interfaces, and performance metrics.
- **Standards**:
  - Provide a set of guidelines and norms that establish uniformity and compatibility across different software components.

    - Define coding conventions, best practices, and technical requirements that developers should follow in their software development process.
    - They promote consistency in coding style, naming conventions, and documentation.
    - Standards also facilitate interoperability among different software components and systems and address security and quality concerns.

Impact of Specifications and Standards

- Specifications and standards contribute to the improved quality and reliability of software systems.
- They enable interoperability and seamless integration with other systems.
- By promoting reusability, modularity, and adherence to best practices, specifications and standards increase productivity and efficiency in software development.

Project Management

- A systematic approach to planning, executing, controlling, and closing projects within an organization.

Why do companies need PM

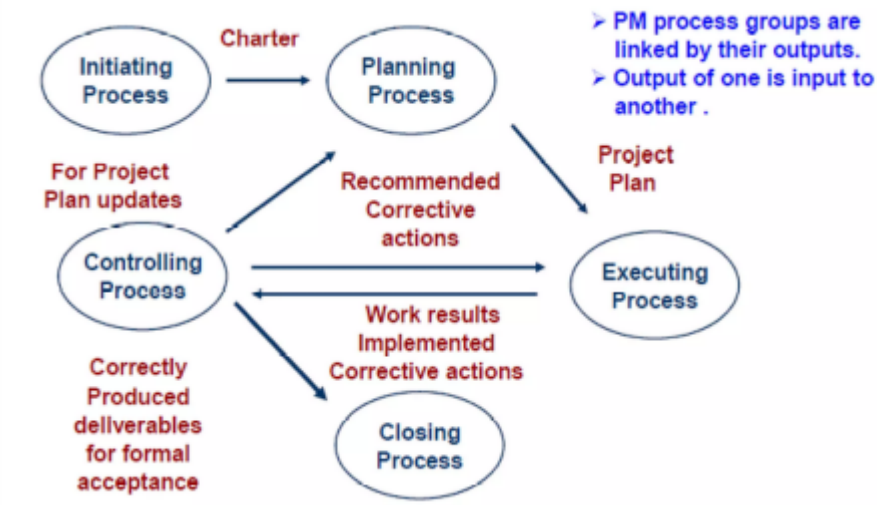
- To handle projects effectively in an organization.
- To plan and assess resource needs for the project
- To estimate project, cost and make proposals
- To plan & schedule activities in a project.
- To assess risk and failure points and make backup plans.
- To lead a project team effectively and communicate well

Why do people learn PM?

- To explore the latest concepts and techniques of project management.
- To increase value/contribution to the organization.
- To prove yourself skillful in managing projects.
- To learn a new thought process that helps organized thinking and structured approach.
- To acquire a professional degree/ recognition and increase job prospects. Endless possibilities and benefits.

Most Important phase of project management

- Planning phase
- Scheduling phase
- Controlling phase
- Closing phase



Project Management Methodologies

- **Waterfall Model**: A linear and sequential approach, where project phases flow downwards like a waterfall.
- **Agile Methodology**: An iterative and incremental approach, emphasizing flexibility and customer satisfaction.
- **Prince2**: A process-driven project management method.
- **Scrum**: A subset of Agile, Scrum is focused on delivering the highest value in the shortest time. It involves dividing the project into sprints, short time frames with a defined list of deliverables.
- **Lean Project Management**: Inspired by lean manufacturing principles, this methodology emphasizes delivering more value with less waste. It focuses on continuous improvement, optimizing processes, and reducing unnecessary expenses.

Quality Assurance

- The totality of features and characteristics of a product or service that meet stated or implied needs.
- **Purpose**: to prevent defects in products and solve problems in delivering services to customers, ensuring products and services meet required quality levels.
- **Key Actions**:
  - Identify QA objectives and standards.
  - Prioritize preventative measures over corrective actions.
  - Collect and analyze relevant data.
  - Establish and monitor quality performance measures.
  - Conduct quality audits.

Key Terms Related to Quality Assurance

- Quality Planning
- Quality Control
- Quality Improvement

Principles of Quality Assurance

- Customer focus
- Leadership
- Involvement of people
- Process approach
- System approach to management
- Continual improvement
- Factual approach to decision making
- Mutually beneficial supplier relationship

Components of Quality Assurance

- Structure Evaluation
- Process Evaluation
- Outcome Evaluation

Levels of Evaluation of Quality

- National Level
- Trust or organization level
- Local Level

Approaches of Quality Improvement

- General Approaches
    - Credentialing
    - Licensure
    - Accreditation
    - Certification
    - Charter
    - Academic Degrees
- Specific Approaches
    - Peer Review Committees (Staff Review Committees)
    - Standard as a device for quality assurance

Integrated Logistics Support (ILS)

- Integrated approach to managing and optimizing logistics throughout a system's lifecycle.
- **Purpose**: to ensure effective, efficient, and economical system support.
- **Objectives**:
    - Enhance system supportability and efficiency.
    - Improve reliability, maintainability, and availability.
    - Reduce lifecycle costs and increase return on investment.

Why Need ILS - Enhances system readiness - Reduces life cycle costs - Improves maintenance efficiency - Ensures supply chain resilience - Facilitates equipment sustainment

Key Components of ILS

- `System Design Integration` : incorporating logistics considerations in design phase.
- `Maintenance Planning` : developing strategies for efficient system maintenance.
- `Supply Chain Management` : optimizing resource and parts availability.

Most widely accepted list of ILS Key Methods

- Reliability Engineering
- Maintainability Engineering
- Maintenance Planning
- Support and Test Equipment
- Manpower and Personnel Training
- Technical Data/Publications Management
- Computer Resources Support
- Facilities Management
- Packaging, Handling, Storage, and Transportation
- Design Interface

Operations

- Operations are practical execution and management of system functionalities.
- It includs tasks such as defining requirements and ensuring system effectiveness in real-world scenarios.
- Operators oversee day-to-day functioning and maintenance of systems, for achieving desired outcomes within the broader framework of systems engineering. They’re the ones who figure out what a system needs to do based on how it’s used.

Key Contributions of Operations Personnel

- **Requirements Definition**: operators translate operational needs into tangible system performance criteria, shaping the entire systems engineering effort.
- **Testing and Validation**: operators actively participate in diverse testing activities to ensure system functionalities align effectively with operational requirements.
- **Operational Insights**: continous feedback from operators ensures that the evolving system aligns seamlessly with real-world operational needs, maintaining operational efficiency throughout development stages.

Software Engineering

- Organized way of making and taking care of computer programs.
- Creating and managing these programs in a planned and efficient manner, ensuring they’re made and updated properly, without taking too long or costing too much.

Hardware Engineering

- The discipline focused on designing, developing, and testing physical components and systems such as computer hardware , circuits & devices.
- Steps:
  - Development process
  - Integration and Testing
  - Collabration With Software Engineering
  - Individualized Management
  - Documentation and Management

Systems Engineering and Development Approaches

- System development refers to the process of creating, designing, and implementing a system to meet specific organizational or user needs.
- **SDLC**:
  - Provides overall framework for managing systems development process.
  - Approaches:
    - `Predicitve approach` :
      - Assumes project can be planned out in advance.
      - Requirements well understood and well defined.
      - Low technical risk
      - Less flexible
    - `Adapative approach` :
      - Assumes project can't be planned out in advance.
      - Requirements and needs uncertain.
      - High technical risk
      - More flexibile
- **Phases in SDLC**:
  - Activities of each "phase" are similar.
  - Phases are not always sequential Phases can overlap.
  - Activities across phases can be done within an iteration.
  - Project planning:
    - Define business problem and scope.
    - Produce detailed project schedule.
    - Confirm project feasibility (Economic, organizational, technical, resource, and schedule).
    - Staff the project (resource management).
    - Launch project official announcement.

- Analysis:
    - Gather information to learn problem domain.
    - Define system requirements Build prototypes for discovery of requirements Prioritize requirements.
    - Generate and evaluate alternatives.
    - Review recommendations with management.
- Design:
    - Design and integrate the network.
    - Design the application architecture.
    - Design the user interfaces.
    - Design the system interfaces.
    - Design and integrate the database.
    - Prototype for design details.
    - Design and integrate system controls.
- Implementation:
    - Construct software components Verify and test
    - Convert data
    - Train users and document the system
    - Install the system
- Models:
    - Waterfall:
        - Each life cycle phase is completed in sequence and then the results of the phase flow on to the next phase.
        - There is no going back once the phase is completed or it’s extremely difficult to do.
        - `Advantages` :
            - Identification of system requirements long before programming.
            - It minimizes changes to the requirements as the project proceeds.
        - `Disadvantages` :
            - The design must be completely specified on paper before programming begins.
            - A long time elapses between the completion of the system proposal in the analysis phase and the delivery of the system.
            - Users rarely are prepared for their introduction to the new system.
            - A paper document is often a poor communication mechanism.
            - If the project team misses important requirements, expensive post-implementation programming may be needed.
            - A system may require significant rework.

- Incremental Development:
  - An iterative approach to software and systems development where the system is built incrementally through a series of smaller, manageable steps.
  - Each iteration involves adding new functionality to the system, allowing for continuous improvement and refinement.
  - `Advantages`：
    - Early and frequent delivery of working software
    - Flexibility and adaptability
    - Risk reduction
    - Stakeholder involvement
  - `Disadvantages`：
    - Coordination and integration challenges
    - Increased complexity
    - Potential for scope creep
    - Resource allocation
- Evolutionary:
  - An approach that emphasizes the iterative and incremental nature of system development.
  - It acknowledges that requirements and constraints often evolve over time, and therefore, the system should be designed to accommodate these changes.
  - `Advantages`：
    - Flexibility and adaptability
    - Early delivery of core functionality
    - Stakeholder involvement
    - Reduced risk
  - `Disadvantages`：
    - Uncertainty and ambiguity
    - Potential for scope creep
    - Resource and time commitment
    - Complexity and maintenance
- Prototyping Development:
  - Performs analysis, design and implementation phases concurrently, and all three phases are performed repeatedly in a cycle until the system is completed.
  - `Advantages`：
    - Very quickly provides a system for users to interact with.
    - It reassures the users that the project team is working on the system.
    - The users can interact with the prototype to better understanding what it can and cannot do rather than attempting to understand a system specification on paper.

- `Disadvantages`：
    - Fast-paced system releases challenge attempts to conduct careful, methodical analysis.
    - Often the prototype undergoes such significant changes that many initial design decisions become poor ones.
    - This can cause problems in the development of complex systems because fundamental issues and problems are not recognized until well into the development process.
- Spiral Development:
    - Breaks each project into smaller pieces, each with a different type of risk (Sources of risk: undefined requirements, complex technology, uncertain competitive environment).
    - `Advantages`：
        - Risk Management
        - Flexibility
        - Stakeholder Involvement
        - Early Prototyping
    - `Disadvantages`：
        - Complexity
        - Resource Intensive
        - Documentation Overhead
        - Control and Monitoring