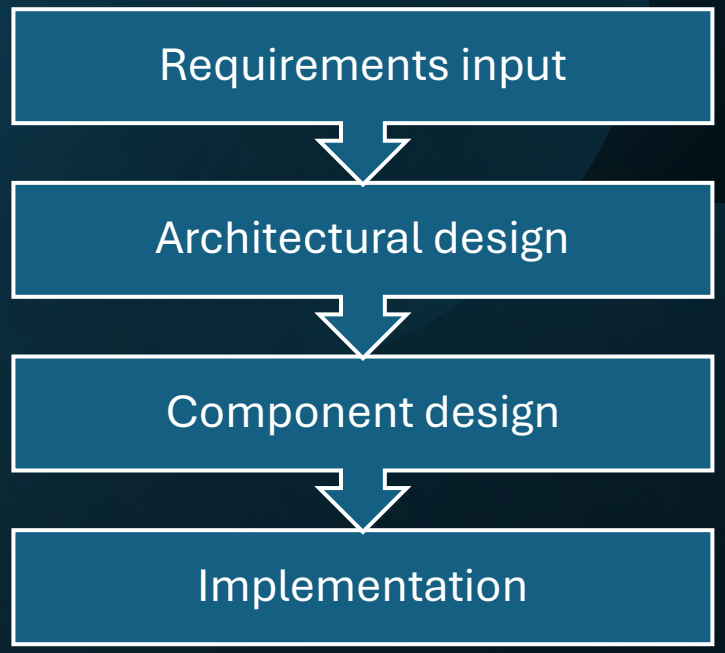
SYS.3 System Architectural Design

Introduction to System Architectural Design



System architecture transforms requirements into actionable design, forming the backbone of development

1. Define System Structure and Elements

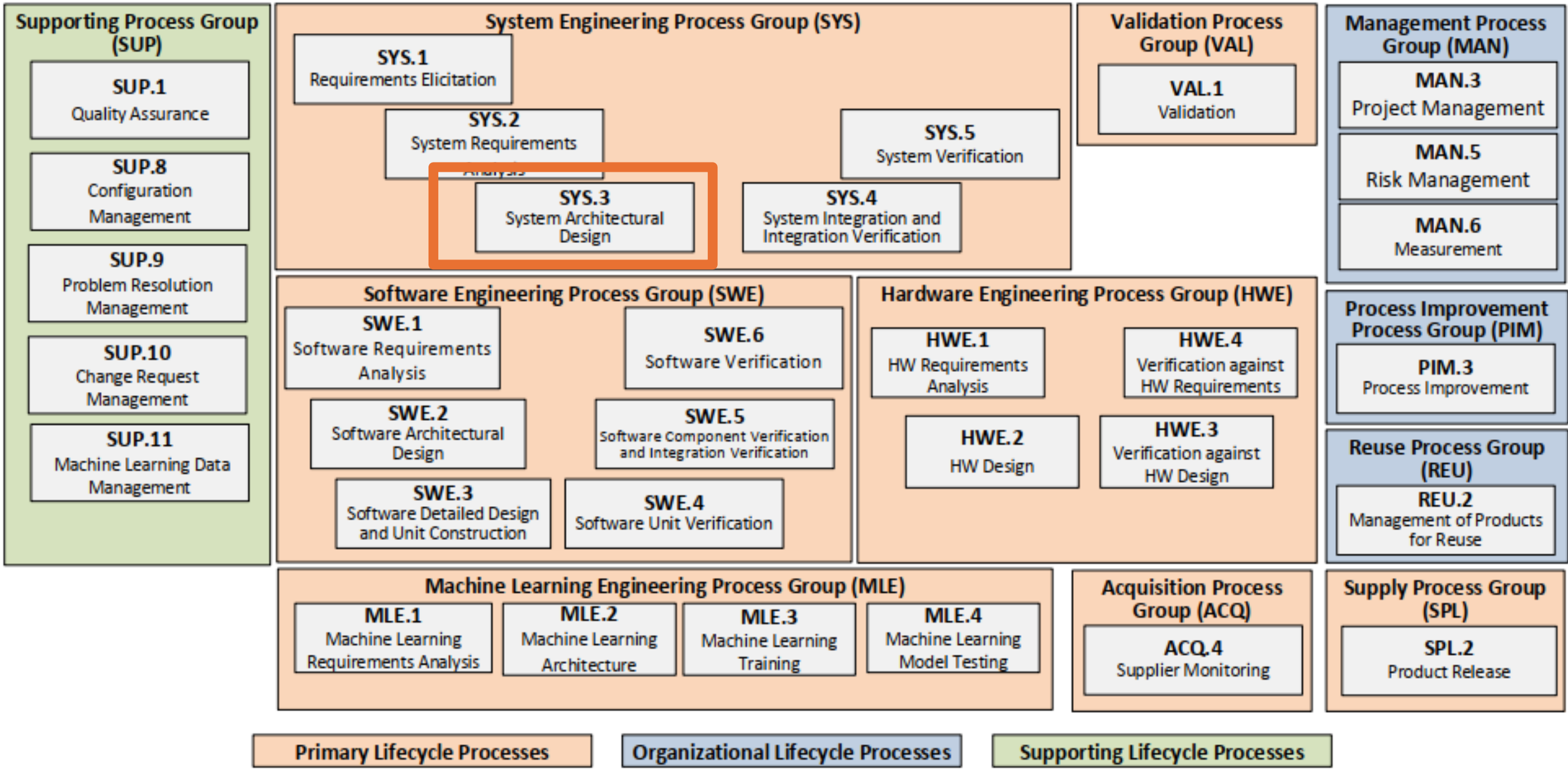
- Identify all necessary components for system functionality.
- Organize elements to ensure a modular and scalable design.
- Establish boundaries for each system component to prevent overlap.

2. Establish Interfaces and Relationships

- Define clear interfaces between system elements to ensure compatibility.
- Detail how components will communicate and interact.
- Outline dependencies among components to avoid integration issues.

3. Ensure Traceability from Requirements

- Map architecture back to initial system requirements.
- Track changes and their impact on the architecture.
- Facilitate seamless tracking from requirements to design.



ASPICE – **A**utomotive **S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination

Process ID

SYS.3

Process name

System Architectural Design

Process purpose

The purpose is to establish an analyzed system architecture consistent with the system requirements.

Process outcomes

1) A system architecture is designed including a definition of the system elements with their behavior, their interfaces, their relationships, and their interactions.
2) The system architecture is analyzed against defined criteria, and special characteristics are identified.
3) Consistency and bidirectional traceability are established between system architecture and system requirements.
4) The agreed system architecture and the special characteristics are communicated to all affected parties.

SYS.3 System Architectural Design	Outcome 1	Outcome 2	Outcome 3	Outcome 4
Output Information Items				
04-06 System Architecture	X			
13-51 Consistency Evidence			X	
13-52 Communication Evidence				X
15-51 Analysis Results		X		
17-57 Special Characteristics		X		
Base Practices				
BP1: Specify static aspects of system architecture	X			
BP2: Specify dynamic aspects of system architecture	X			
BP3: Analyze the system architecture		X		
BP4: Ensure consistency and establish bidirectional traceability			X	
BP5: Communicate agreed system architecture				X

Base practices

SYS.3.BP1: Specify static aspects of the system architecture. Specify and document the static aspects of the system architecture with respect to the functional and nonfunctional system requirements, including external interfaces and a defined set of system elements with their interfaces and relationships.

SYS.3.BP2: Specify dynamic aspects of the system architecture. Specify and document the dynamic aspects of the system architecture with respect to the functional and nonfunctional system requirements including the behavior of the system elements and their interaction in different system modes.

Note 1: Examples of interactions of system elements are timing diagrams reflecting inertia of mechanical components, processing times of ECUs, and signal propagation times of bus systems.

SYS.3.BP3: Analyze system architecture. Analyze the system architecture regarding relevant technical design aspects related to the product lifecycle, and to support project management regarding project estimates, and derive Special Characteristics for hardware elements. Document a rationale for the system architectural design decision.

Note 2: See MAN.3.BP3 for project feasibility and MAN.3.BP5 for project estimates.

Note 3: Examples for product lifecycle phases are production, maintenance & repair, decommissioning.

Note 4: Examples for technical aspects are manufacturability for production, suitability of pre-existing system elements to be reused, or availability of system elements.

Note 5: Examples for methods being suitable for analyzing technical aspects are prototypes, simulations, and qualitative analyses (e.g. FMEA approaches)

Note 6: Examples of design rationales are proven-in-use, reuse of a product platform or product line), a make-or-buy decision, or found in an evolutionary way (e.g. set-based design).

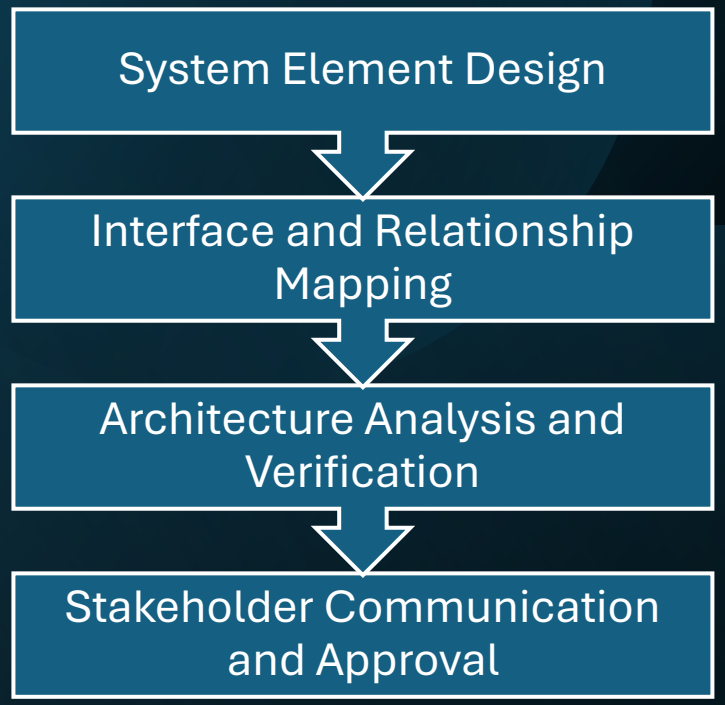
SYS.3.BP4: Ensure consistency and establish bidirectional traceability. Ensure consistency and establish bidirectional traceability between the elements of the system architecture and the system requirements that represent properties or characteristics of the physical end product.

Note 7: Bidirectional traceability further supports consistency, and facilitates impact analysis of change requests, and demonstration of verification coverage.

Note 8: There may be non-functional system requirements that the system architectural design does not trace to. Examples are development process requirements. Such requirements are still subject to verification.

SYS.3.BP5: Communicate agreed system architecture. Communicate the agreed system architecture, including the special characteristics, to all affected parties.

SYS.3 System Architectural Design Process Overview



SYS.3 process ensure that architecture design is robust, traceable, and aligned with Stakeholder needs

1. Design System Elements and Interfaces

- Ensure each component's purpose and interfaces are clear.
- Design is modular, supporting scalability.
- Interfaces facilitate seamless data flow and functionality.

2. Analyze Architecture against Quality Criteria

- Perform assessments for reliability, scalability, and maintainability.
- Identify any special characteristics (e.g., redundancy).
- Address potential risks to enhance robustness.

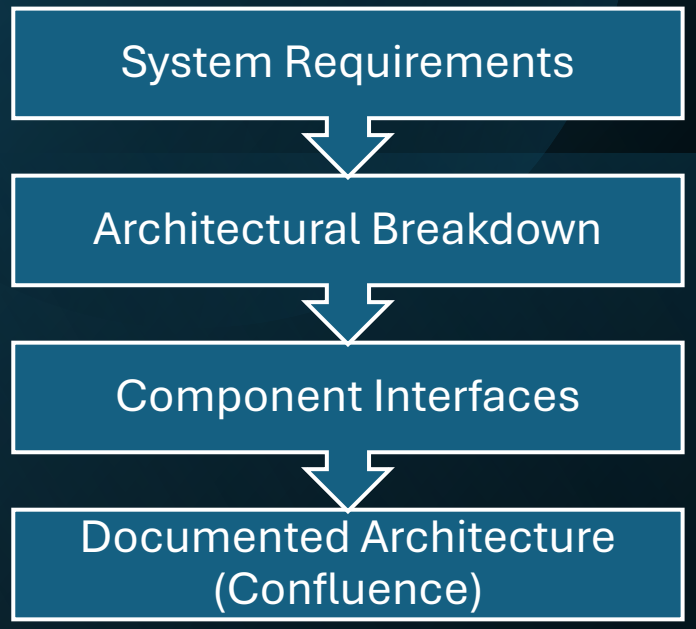
3. Establish Traceability

- Link architectural elements to system requirements.
- Support bidirectional traceability to maintain alignment.
- Facilitate impact analysis when changes are needed.

4. Communicate Architecture

- Ensure stakeholders understand the architecture.
- Address stakeholder questions and gain approval.
- Document architecture details for future reference.

BP1: Specify Static Aspects of the System Architecture



Breaking down the system into structured modules simplifies integration and clarifies component roles

SYS.3.BP1: Specify static aspects of the system architecture. Specify and document the static aspects of the system architecture with respect to the functional and nonfunctional system requirements, including external interfaces and a defined set of system elements with their interfaces and relationships.

1. Identify Core Components and Modules

- Break down the system into main components (e.g., UI, processing, data storage).
- Outline each component's purpose in the architecture.
- Ensure components align with system requirements.

2. Define External Interfaces

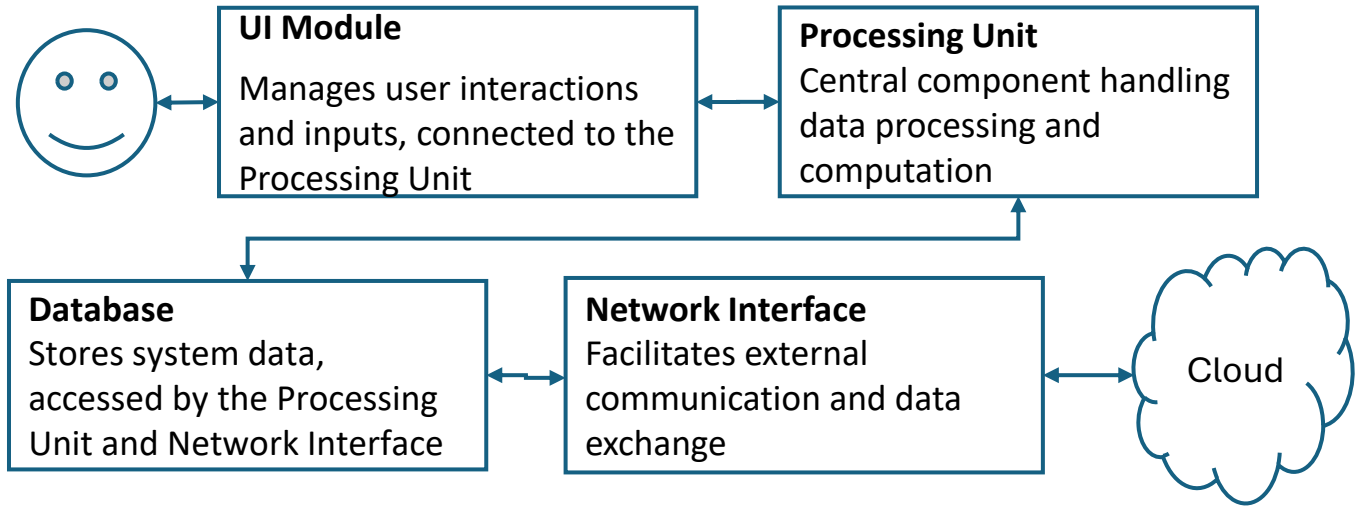
- List external systems interacting with the architecture.
- Specify data flows and protocols for each interface.
- Ensure external interfaces are compatible with industry standards.

3. Map Relationships and Dependencies

- Establish links between system components.
- Document any dependencies for clear integration.
- Outline data flow and control between components.

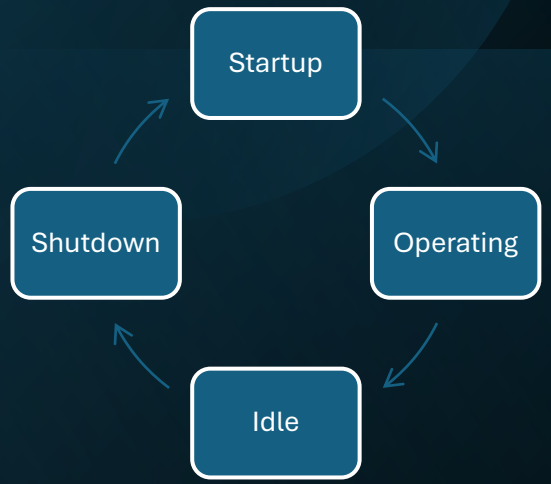
BP1 Example: High-Level System Architecture Diagram

A static component diagram details the system's core structure, showing components, external interfaces, and internal relationships.



Component	Description	Interacts With
User (UI Input)	Source of user interactions and inputs	UI Module
UI Module	Manages user interactions and inputs, connected to Processing	User, Processing Unit
Processing Unit	Central component handling data processing and computation	UI Module, Database, Network Interface
Database	Stores system data, accessed by Processing Unit and Network	Processing Unit, Network Interface
Network Interface	Facilitates external communication and data exchange with cloud	Database, Processing Unit, Cloud
Cloud	Provides external services and data storage	Network Interface

BP2: Specify Dynamic Aspects of the System Architecture



The dynamic aspects address how system components behave in different modes and conditions

SYS.3.BP2: Specify dynamic aspects of the system architecture. Specify and document the dynamic aspects of the system architecture with respect to the functional and nonfunctional system requirements including the behavior of the system elements and their interaction in different system modes.

1. Define Behavioral Interactions

- Map interactions between components under normal operation.
- Identify control flows and response patterns.
- Establish operational dependencies and protocols.

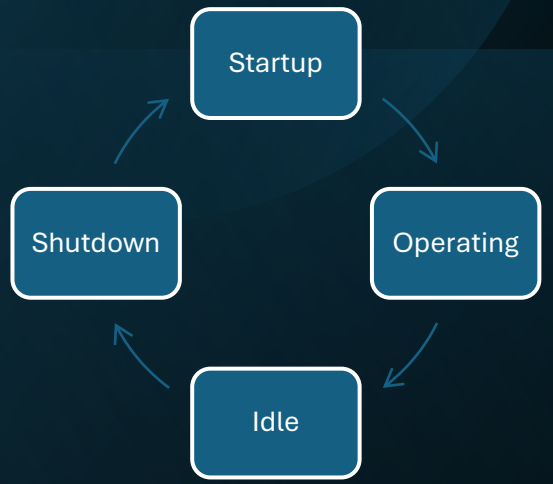
2. Document System Modes

- Specify different operating modes (e.g., startup, shutdown, idle).
- Outline behavior and constraints for each mode.
- Ensure modes support system objectives (e.g., energy-saving mode).

3. Describe State Transitions

- Hold review sessions with stakeholders to ensure their expectations are understood.
- Verify that all requirements are feasible and clear.
- Obtain confirmation from stakeholders that their needs are accurately captured.

BP2 Example: Specify Dynamic Aspects of the System Architecture



A timing and sequence diagram captures the real-time interactions and transitions between system components in operation

1. Initialization Mode (Startup)

- Time 0: System startup begins.
- Time 1: UI Module activates and sends initial data to the Processing Unit.
- Time 2: Processing Unit performs startup checks and loads initial data from Database.
- Time 3: Network Interface establishes connection with the Cloud.

2. Operational Mode

- Time 4: User sends input via UI Module.
- Time 5: UI Module transmits user data to Processing Unit.
- Time 6: Processing Unit processes data and retrieves additional information from Database.
- Time 7: Processing Unit sends processed data to Network Interface for external communication.
- Time 8: Network Interface exchanges data with the Cloud.

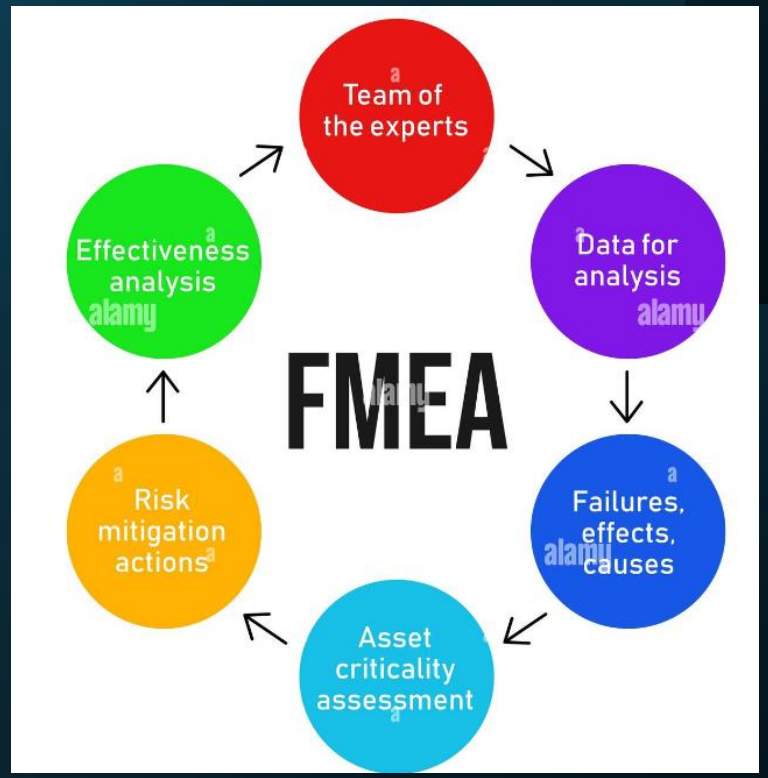
3. Idle Mode

- Time 9: No user input; system components are in standby.
- Time 10: Processing Unit periodically saves session data to Database.
- Time 11: Network Interface checks for updates from Cloud.

1. Shutdown Mode

- Time 12: System shutdown initiated by Processing Unit.
- Time 13: UI Module disconnects and saves settings.
- Time 14: Processing Unit performs final save to Database.
- Time 15: Network Interface disconnects from Cloud, system powers down.

BP3: Analyze System Architecture



An FMEA chart helps identify and mitigate potential failure points in the architecture, enhancing overall robustness.

> **SYS.3.BP3: Analyze system architecture.** Analyze the system architecture regarding relevant technical design aspects related to the product lifecycle, and to support project management regarding project estimates, and derive Special Characteristics for hardware elements. Document a rationale for the system architectural design decision.

1. Evaluate Lifecycle Compatibility

- Assess architecture across lifecycle stages (production, maintenance, end-of-life).
- Ensure design supports scalability, upgrades, and decommissioning.
- Use lifecycle analysis to predict maintenance and resource needs.

2. Determine Technical Feasibility

- Conduct feasibility checks on system components.
- Use Failure Mode and Effects Analysis (FMEA) to identify potential failure points.
- Validate technical assumptions with prototypes or simulations.

https://en.wikipedia.org/wiki/Failure_mode_and_effects_analysis

3. Identify Special Characteristics

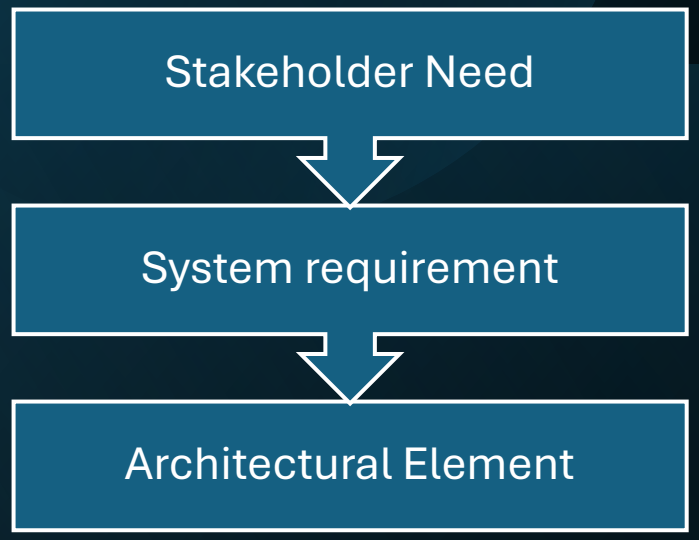
- Flag components requiring high reliability or performance.
- Define critical attributes like fault tolerance and redundancy.
- Plan additional testing or quality measures for key areas

BP3 Example: System Architecture Analysis

An FMEA table helps document potential failure modes, their associated risks, and recommended mitigation strategies for critical system components

Component	Potential Failure Mode	Risk Level	Mitigation Strategy
User	Incorrect input or accidental actions	Medium	Provide input validation, add undo/confirm options
UI Module	User input not registering	Medium	Add error handling for input, test for responsiveness
Processing Unit	Overheating during data processing	High	Install temperature monitoring, implement cooling system
Database	Data corruption from unexpected shutdown	Medium	Enable regular backups, implement redundancy
Network Interface	Loss of external connectivity	High	Add reconnection logic, use failover networks
Cloud	Service downtime or slow response	Medium	Implement local caching, add retry mechanisms

BP4: Ensure Consistency and Establish Bidirectional Traceability



A traceability matrix links architectural elements to requirements, ensuring comprehensive coverage and facilitating change management.

SYS.3.BP4: Ensure consistency and establish bidirectional traceability. Ensure consistency and establish bidirectional traceability between the elements of the system architecture and the system requirements that represent properties or characteristics of the physical end product.

Note 7: Bidirectional traceability further supports consistency, and facilitates impact analysis of change requests, and demonstration of verification coverage.

Note 8: There may be non-functional system requirements that the system architectural design does not trace to. Examples are development process requirements. Such requirements are still subject to verification.

1. Establish Traceability Links

- Link each architectural component to relevant requirements.
- Use a traceability matrix to map connections.
- Regularly update links to reflect changes in design or requirements

2. Bidirectional Traceability

- Ensure forward and backward traceability.
- Confirm that all requirements are addressed in the architecture.
- Support impact analysis when changes occur.

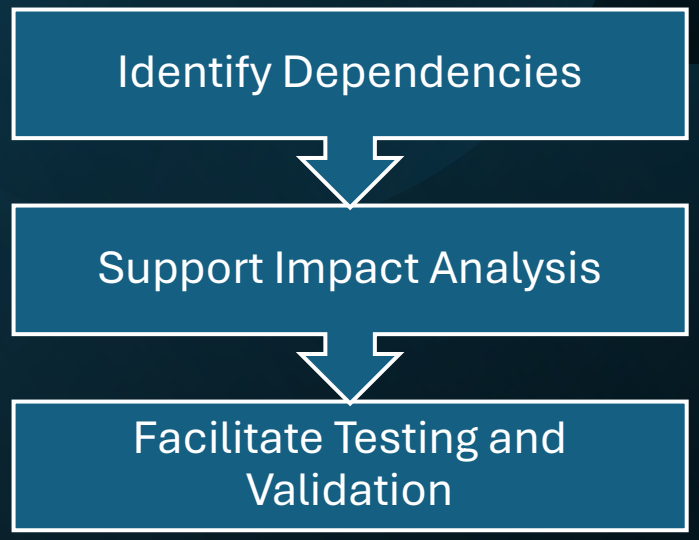
3. Verification Support for All Requirement Types

- Facilitate testing by linking design elements to requirements.
- Provide verification coverage for non-functional and process requirements.
- Ensure verification of requirements that don’t directly map to architecture.

BP4 Example: System Architecture Traceability Matrix

A traceability matrix provides a clear, visual representation of connections between architectural components and requirements, supporting easy tracking and verification.

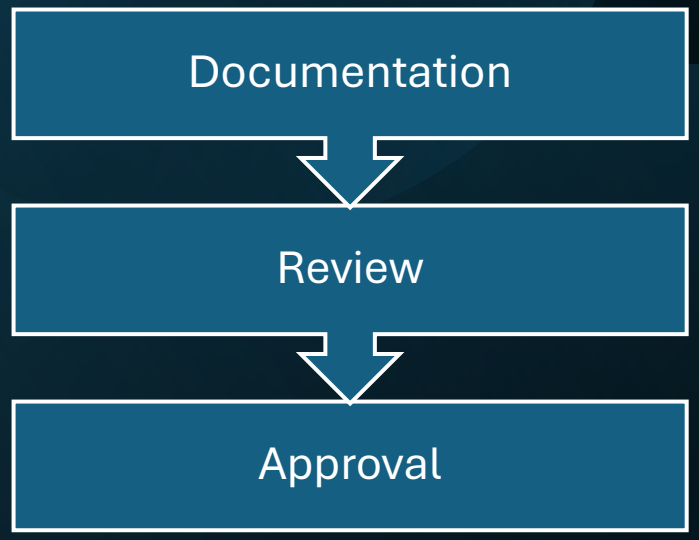
System Requirements	UI Module	Processing Unit	Database	Network Interface	Cloud
System Requirement 1: The system shall process user inputs within 1 second to ensure a responsive user experience.	✅	✅			
System Requirement 2: The system shall handle up to 100 transactions per second to meet performance targets.		✅	✅		
System Requirement 3: The system shall store up to 1 TB of data to accommodate long-term data storage requirements.			✅	✅	
System Requirement 4: The system shall ensure secure data transmission for external communication to maintain data integrity and confidentiality.				✅	✅
System Requirement 5: The system shall support secure access to external APIs to enable secure data exchange with third-party services.				✅	✅



Explanation of Requirements and Consistency

1. **System Requirement 1**: Linked to **UI Module** and **Processing Unit**
    1. *The system shall process user inputs within 1 second to ensure a responsive user experience.*
    2. This requirement impacts both the **UI Module** (for capturing inputs) and the **Processing Unit** (for processing inputs in real-time).
2. **System Requirement 2**: Linked to **Processing Unit** and **Database**
    1. *The system shall handle up to 100 transactions per second to meet performance targets.*
    2. This requirement affects both the **Processing Unit** (for processing transactions) and the **Database** (for storing transaction data efficiently).
3. **System Requirement 3**: Linked to **Database** and **Network Interface**
    1. *The system shall store up to 1 TB of data to accommodate long-term data storage requirements.*
    2. This requirement involves the **Database** for storage and the **Network Interface** for data exchange or backup purposes.
4. **System Requirement 4**: Linked to **Network Interface** and **Cloud**
    1. *The system shall ensure secure data transmission for external communication to maintain data integrity and confidentiality.*
    2. This requirement impacts the **Network Interface** (for data transmission) and **Cloud** (for secure storage and interaction with external services).
5. **System Requirement 5**: Linked to **Network Interface** and **Cloud**
    1. *The system shall support secure access to external APIs to enable secure data exchange with third-party services.*
    2. This requirement involves the **Network Interface** for API interactions and **Cloud** for data exchange security.

BP5: Communicate Agreed System Architecture



Communication engage stakeholders, address feedback, and obtain formal approval for the system architecture

SYS.3.BP5: Communicate agreed system architecture. Communicate the agreed system architecture, including the special characteristics, to all affected parties.

1. Document Architecture Clearly

- Use diagrams, tables, and detailed explanations.
- Create documentation that is accessible to technical and non-technical stakeholders.
- Provide a reference for future design updates.

2. Engage Stakeholders in Review

- Organize review sessions to explain the architecture.
- Address questions and clarify design decisions.
- Encourage feedback for improvements.

3. Formalize Approval

- Obtain formal approval from stakeholders.
- Ensure architecture meets business and technical needs.
- Establish the architecture baseline for development.

BP5 Example : Stakeholder Communication Plan

A communication plan outlines how and when to communicate the system architecture to stakeholders, ensuring regular updates and alignment

Communication Activity	Audience	Frequency	Format
Project Review Meetings	Project Team, Stakeholders	Bi-weekly	In-person/Online meeting
Architecture & Documentation	Dev Team, QA Team	Monthly	Document repository
Feedback & Input Collection	Stakeholders, End-users	Monthly	Surveys, Feedback forms
Status & Change Updates	Project Team, Management	Weekly	Email, Dashboard updates
Final Project Presentation	All Stakeholders	End of project	In-person/Online presentation

Challenges in System Architectural Design

Major challenges in System Architectural Design provides key areas to focus on for overcoming them

1. Balancing Flexibility and Scalability ⚖️

- Challenge of creating an architecture that accommodates growth without becoming overly complex.
- Requires strategic planning to ensure adaptability without excessive resource use.
- Ensures long-term maintainability.

2. Ensuring Performance and Reliability 🚀

- Managing trade-offs between high performance and reliable functionality.
- Essential to align architecture with performance requirements under real-world conditions.
- Testing for robustness and responsiveness is critical.

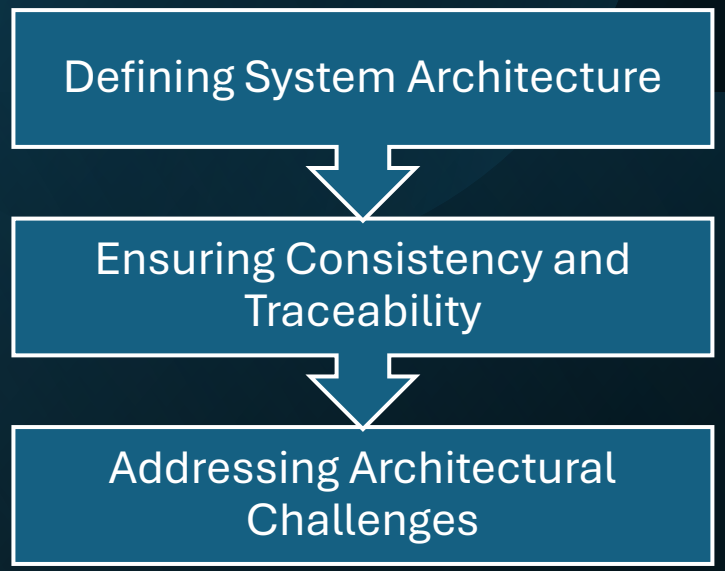
3. Integration of Components and Interfaces 🔗

- Complexity in integrating diverse components, each with unique specifications and protocols.
- Requires consistent interface management to avoid compatibility issues.
- Seamless interaction between components is essential for system cohesion.

4. Maintaining Security and Compliance 🔒

- Architectural design must meet security standards and regulatory requirements.
- Challenge of protecting data integrity while allowing necessary accessibility.
- Ensures the system meets legal and operational guidelines.

Summary and Q&A



Technical and strategic aspects are essential to building a well-structured system

1. Defining System Architecture

- Established the purpose and components of a robust system architecture.
- Emphasized both static and dynamic aspects in line with requirements.
- Focused on building a scalable and adaptable design framework.

2. Ensuring Consistency and Traceability

- Highlighted the importance of bidirectional traceability between architecture and requirements.
- Addressed alignment with system requirements for seamless integration.
- Discussed methods to ensure architecture remains cohesive and aligned with stakeholder goals.

3. Addressing Architectural Challenges

- Explored challenges related to scalability, complexity, and performance.
- Focused on integrating diverse components within the architecture.
- Considered strategies for managing flexibility without compromising system efficiency.