What is the UAP Reference Guide?

The **Unified Architecture Process (UAP)** is a comprehensive, production-ready methodology for cost-effective development of high-quality software architectures. It provides a structured framework that encompasses all key design activities and offers clear, step-by-step implementation guidance.

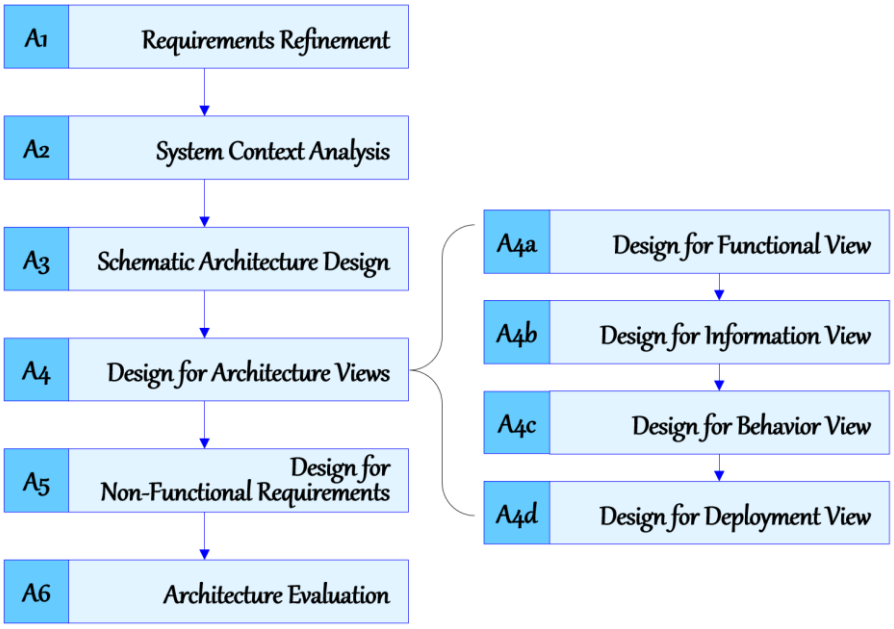
UAP Reference Guide provides a concise overview of the Unified Architecture Process. It summarizes the key activities, outlines the steps within each activity, and highlights the tasks associated with each step. The descriptions are intentionally brief to serve as a quick and accessible reference.

Process Model of UAP

The UAP is structured into three hierarchical work units: activities, steps, and tasks.

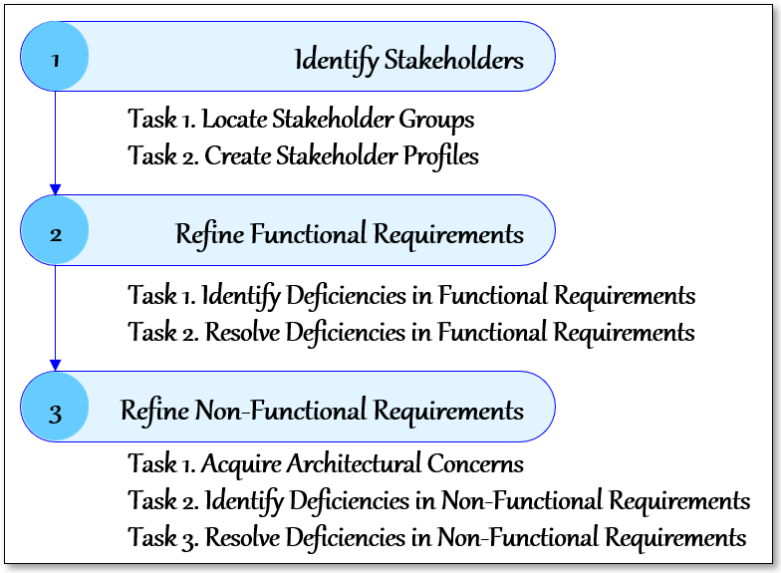
- An Activity is a high-level unit of work in the architecture design process, representing a major phase with defined objectives and multiple steps.
- A Step is a structured subunit of an activity that outlines actions or procedures and consists of tasks that achieve its objective.
- A Task is the most granular unit of work, representing specific actions required to complete a step.

The UAP process model consists of six main activities (A1–A6), with Activity A4 further refined into design activities for multiple views: functional, information, behavior, and deployment.



Activity A1. Requirements Refinement

The purpose of this activity is to refine the initial requirements provided by stakeholders to ensure clarity and completeness for subsequent design stages. The activity is structured into three steps.



✦ Step 1. Identify Stakeholders

This step is to identify the stakeholders of the target system. Engaging relevant stakeholders enables architects to make more informed and comprehensive design decisions, ensuring that the architecture aligns with stakeholder expectations. This step is carried out through the following tasks.

- Task 1, *Locate Stakeholder Groups*, is to identify key stakeholder groups for the target system, ensuring their involvement in providing insights, expressing concerns, and specifying requirements relevant to the system.
- Task 2, *Create Stakeholder Profiles*, involves identifying specific individuals within each stakeholder group and creating profiles for them. Each stakeholder profile includes details such as identification, contact information, and availability.

✦ Step 2. Refine Functional Requirements

This step is to refine the system's functional requirements by applying the principles and techniques of requirement engineering. It is carried out through the following tasks.

- Task 1, *Identify Deficiencies in Functional Requirements*, is to identify gaps and weaknesses in the functional requirements by thoroughly reviewing each requirement in the initial SRS.
- Task 2, *Resolve Deficiencies in Functional Requirements*, is to address identified issues by clarifying ambiguities with stakeholders, gathering additional insights, and refining the updated functional requirements.

✦ Step 3. Refine Non-Functional Requirements

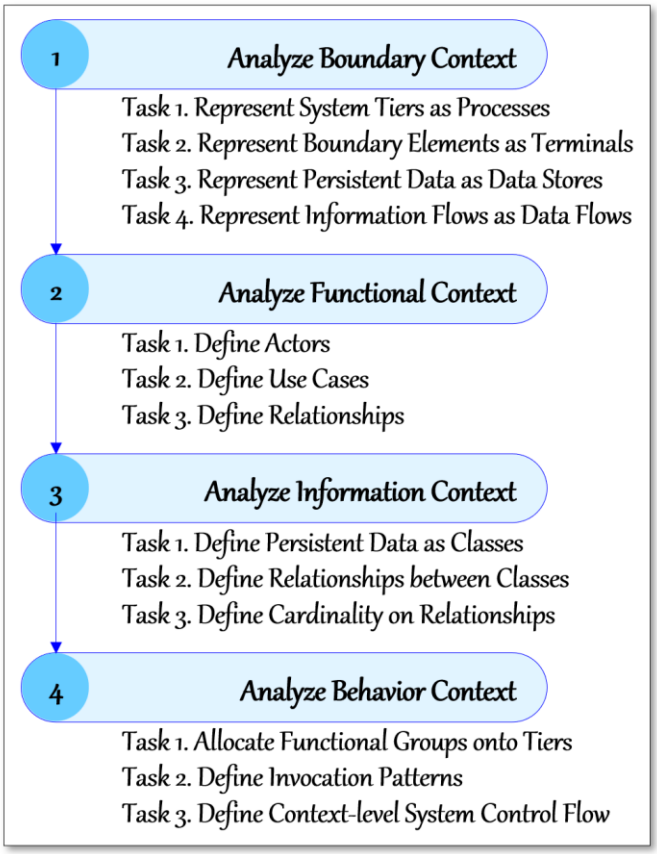
This step is to refine the non-functional requirements of the system by applying principles and techniques of requirement engineering. It is carried out through the following tasks.

- Task 1, *Acquire Architectural Concerns*, is to gather architectural concerns from stakeholders and derive corresponding non-functional requirements (NFRs) based on those concerns.
- Task 2, *Identify Deficiencies in Non-Functional Requirements*, is to detect shortcomings in the NFRs by thoroughly reviewing each requirement in the initial Software Requirements Specification (SRS), using methods similar to those applied for functional requirements.

- Task 3, *Resolve Deficiencies in Non-Functional Requirements*, is to address and resolve identified NFR deficiencies by analyzing each issue, consulting stakeholders for clarification, and revising the requirements accordingly.

ACTIVITY A2. SYSTEM CONTEXT ANALYSIS

The purpose of this activity is to develop a context model for the system by analyzing the refined requirements, where the context model represents a high-level abstraction of the system's boundaries, functionality, datasets, and behavior. The activity is structured into four steps.



✦ Step 1. Analyze Boundary Context

This step is to analyze the boundary context of the target system, which can be effectively represented using a DFD. It is carried out through the following tasks.

- Task 1, *Represent System Tiers as Processes*, is to identify the system tiers of the target system and represent them as processes. In a Level 0 DFD, each process symbolizes a physical system tier, which may correspond to the entire computer system or a specific subsystem.
- Task 2, *Represent Boundary Elements as Terminals*, is to model the boundary elements of the system and represent them as terminals. In a DFD, a terminal represents a boundary element that interacts with the system by providing input or receiving output.
- Task 3, *Represent Persistent Data as Data Stores*, is to identify the persistent datasets managed by the target system and represent them as data stores.
- Task 4, *Represent Information Flows as Data Flows*, is to identify the data flows among processes, terminals, and data stores within the context of the system.

✦ Step 2. Analyze Functional Context

This step is to analyze the functional context of the target system and represent it using a use case diagram. It is carried out through the following tasks.

- Task 1, *Define Actors*, is to identify the actors that interact with the target system, where each actor represents a distinct role assumed by an entity engaging with the system.
- Task 2, *Define Use Cases*, is to analyze the system's functionality and represent it through use cases, each capturing a specific and cohesive functional behavior.
- Task 3, *Define Relationships*, is to define the relationships depicted in the use case diagram, including those between actors, between actors and use cases, and among use cases.

✦ Step 3. Analyze Information Context

This step is to analyze the informational context of the target system and represent it using a class diagram. It is carried out through the following tasks.

- Task 1, *Define Persistent Data as Classes*, is to identify the persistent datasets managed by the system and represent them as persistent object classes.
- Task 2, *Define Relationships between Classes*, is to establish relationships among the identified classes using the five standard relationship types in a class diagram: Dependency, Association, Aggregation, Composition, and Inheritance.
- Task 3, *Define Cardinality on Relationships*, is to specify the cardinalities of relationships, indicating the number of instances of one class that can be associated with instances of another class.

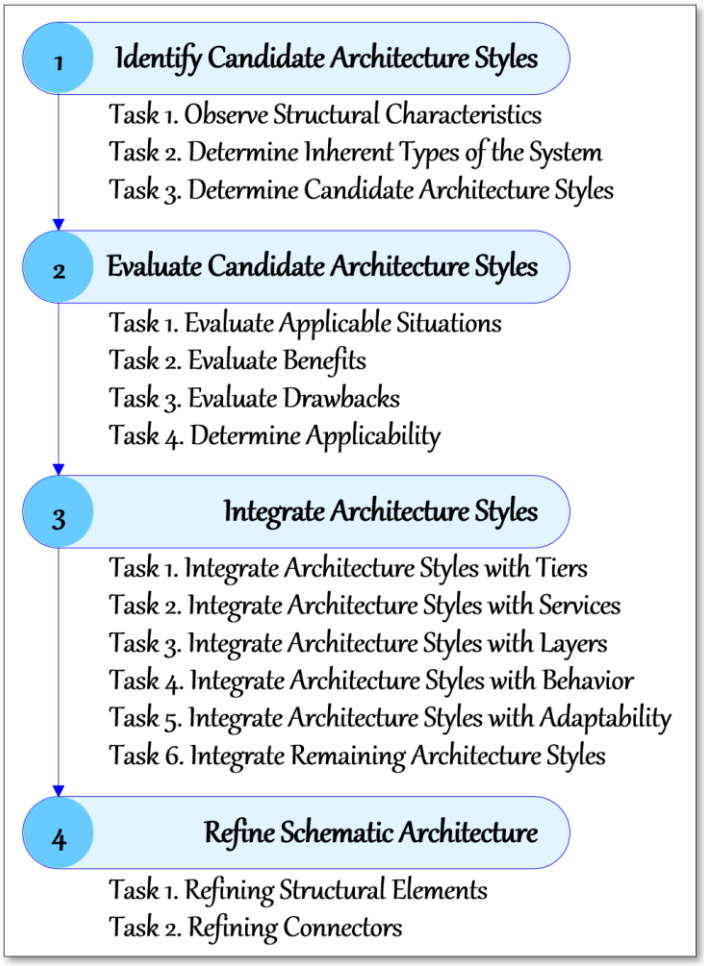
✦ Step 4. Analyze Behavior Context

This step aims to analyze the behavioral context of the target system and represent it using an activity diagram. It is carried out through the following tasks.

- Task 1, *Allocate Functional Groups onto Tiers*, is to assign the functional groups of the system across multiple tiers, aiming to optimally distribute functional components so that each tier is responsible for specific aspects of the system's overall functionality.
- Task 2, *Define Invocation Patterns*, is to identify the most appropriate invocation patterns for each functional group within its designated tier.
- Task 3, *Define Context-level System Control Flow*, is to establish the context-level control flow of the system by creating an activity diagram for each tier, utilizing the invocation patterns associated with the functional groups.

ACTIVITY A3. SCHEMATIC ARCHITECTURE DESIGN

The purpose of this activity is to design the schematic architecture of the target system by integrating appropriate architecture styles. The schematic architecture represents the system's stable and enduring structural layout, serving as the foundation for subsequent architectural decisions. The activity is structured into four steps.



✦ Step 1. Identify Candidate Architecture styles

This step is to identify architecture styles suitable for designing the schematic architecture. It is carried out through the following tasks.

- Task 1, *Observe Structural Characteristics*, is to identify the structural properties and constraints of the target system that are relevant to informing its architectural design.
- Task 2, *Determine Inherent Types of the System*, is to identify the inherent types of the target system in order to narrow down and guide the selection of appropriate architecture styles.
- Task 3, *Determine Candidate Architecture Styles*, is to identify the most suitable architecture styles based on the system's inherent types and structural characteristics.

✦ Step 2. Evaluate Candidate Architecture styles

This step is to evaluate candidate architecture styles to determine their suitability for the target system. It is carried out through the following tasks.

- Task 1, *Evaluate Applicable Situations*, is to assess whether the target system aligns with the applicable conditions and assumptions defined for each candidate architecture style.
- Task 2, *Evaluate Benefits*, is to evaluate the extent to which the target system can leverage the benefits offered by each candidate architecture style.

- Task 3, *Evaluate Drawbacks*, is to analyze whether the target system can effectively manage or mitigate the potential drawbacks associated with each candidate architecture style.
- Task 4, *Determine Applicability*, is to determine the overall suitability of each candidate architecture style by synthesizing the evaluations of applicability, benefits, and drawbacks.

✦ Step 3. Integrate Architecture Styles

This step is to incorporate the selected architecture styles into the schematic architecture of a target system. It is carried out through the following tasks.

- Task 1, *Integrate Architecture Styles with Tiers*, is to incorporate architecture styles that define structural organization in terms of system tiers.
- Task 2, *Integrate Architecture Styles with Services*, is to incorporate architecture styles that utilize external services delivered over a network.
- Task 3, *Integrate Architecture Styles with Layers*, is to incorporate architecture styles that follow a layered architectural approach.
- Task 4, *Integrate Architecture Styles with Behavior*, is to incorporate architecture styles that specify behavioral structures, such as invocation or interaction paradigms.
- Task 5, *Integrate Architecture Styles with Adaptability*, is to incorporate architecture styles that enhance the adaptability and flexibility of the target system.
- Task 6, *Integrate Remaining Architecture Styles*, is to incorporate any remaining architecture styles that have not yet been addressed in the preceding integration tasks.

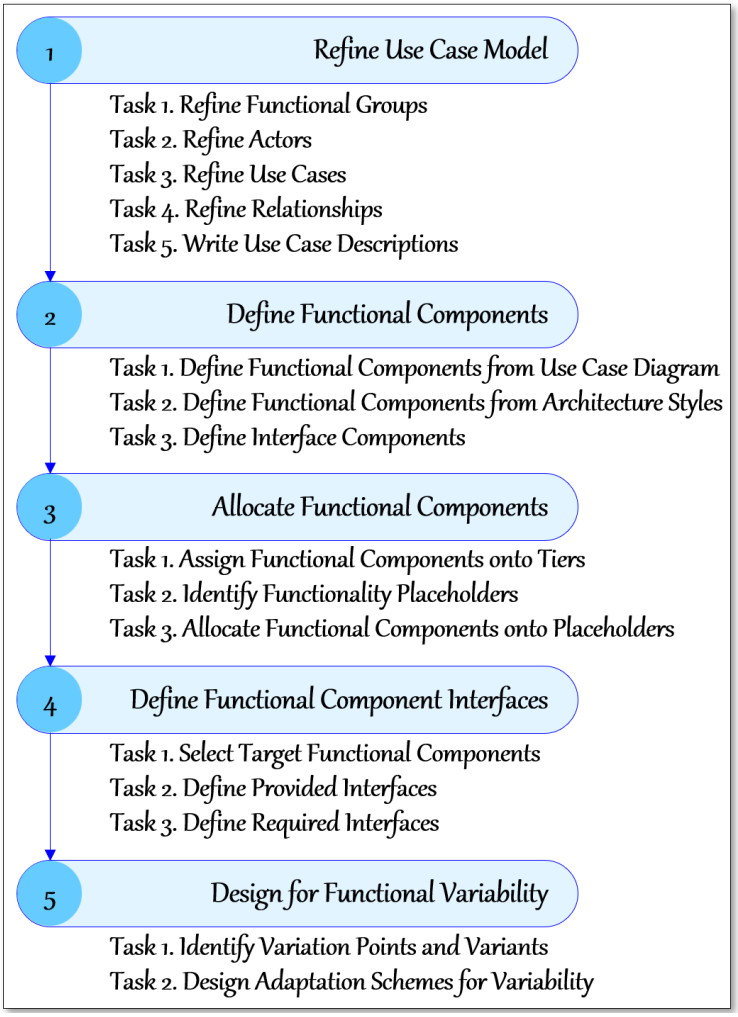
✦ Step 4. Refine Schematic Architecture

This step is to refine the schematic architecture by making additions and modifications to ensure it fully meets the requirements of the target system. It is carried out through the following tasks.

- Task 1, *Refine Structural Elements*, is to further develop and clarify the structural elements of the schematic architecture to ensure completeness and coherence.
- Task 2, *Refine Connectors*, is to refine the connectors within the schematic architecture, detailing the interaction mechanisms and communication paths among components.

ACTIVITY A4A. DESIGN FOR FUNCTIONAL VIEW

The purpose of this activity is to design the architecture for the functional view, which focuses on modeling system functionality, identifying functional components, defining their interfaces, and designing the components while accounting for functional variability. This activity consists of five steps.



Step 1. Refine Use Case Model

This step is to refine the context-level use case diagram and provide detailed descriptions for use cases with complex functionality. It is carried out through the following tasks.

- Task 1, *Refine Functional Groups*, is to refine the functional groups initially identified in the context model to enhance their clarity and alignment with system functionality.
- Task 2, *Refine Actors*, is to refine the actors represented in the context-level use case diagram, ensuring accurate depiction of their roles and interactions with the system.
- Task 3, *Refine Use Cases*, is to refine the use cases in the context-level use case diagram to better capture the system's functional behavior.
- Task 4, *Refine Relationships*, is to refine the relationships in the context-level use case diagram, including associations between actors and use cases as well as dependencies among use cases.
- Task 5, *Write Use Case Descriptions*, is to select use cases that exhibit high complexity, critical functionality, or user-centered behavior, and to document detailed descriptions for each selected use case.

Step 2. Define Functional Components

This step is to define the functional components of the target system based on the use case diagram, by considering the cohesiveness of functionality. It is carried out through the following tasks.

- Task 1, *Define Functional Components from Use Case Diagram*, is to identify functional components by grouping related use cases into cohesive units that represent distinct functionalities of the system.
- Task 2, *Define Functional Components from Architecture Styles*, is to define functional components based on the structural and organizational principles of the selected architecture style.
- Task 3, *Define Interface Components*, is to identify the interface components of the target system, where each interface component specifies the protocols and mechanisms for external or inter-component interactions.

Step 3. Allocate Functional Components

This step is to allocate the function components to their respective functionality placeholders within a schematic architecture. It is carried out through the following tasks.

- Task 1, *Assign Functional Components onto Tiers*, is to allocate each functional component to one or more tiers within the schematic architecture, based on its role and functional scope.
- Task 2, *Identify Functionality Placeholders*, is to identify functionality placeholders within the schematic architecture that represent designated locations for hosting functional components.
- Task 3, *Allocate Functional Components onto Placeholders*, is to assign the functional components to their corresponding functionality placeholders, ensuring consistency with the architectural structure and design intent.

Step 4. Define Functional Component Interfaces

This step is to specify the interfaces of selected functional components. It is carried out through the following tasks.

- Task 1, *Select Target Functional Components*, is to identify the functional components that require a precise and rigorous definition of their interfaces.
- Task 2, *Define Provided Interface*, is to define the provided interface of black-box functional components, where the provided interface specifies the set of services offered to other components.
- Task 3, *Define Required Interface*, is to define the required interface for functional components that rely on services offered by other components, specifying the methods and interactions needed for integration.

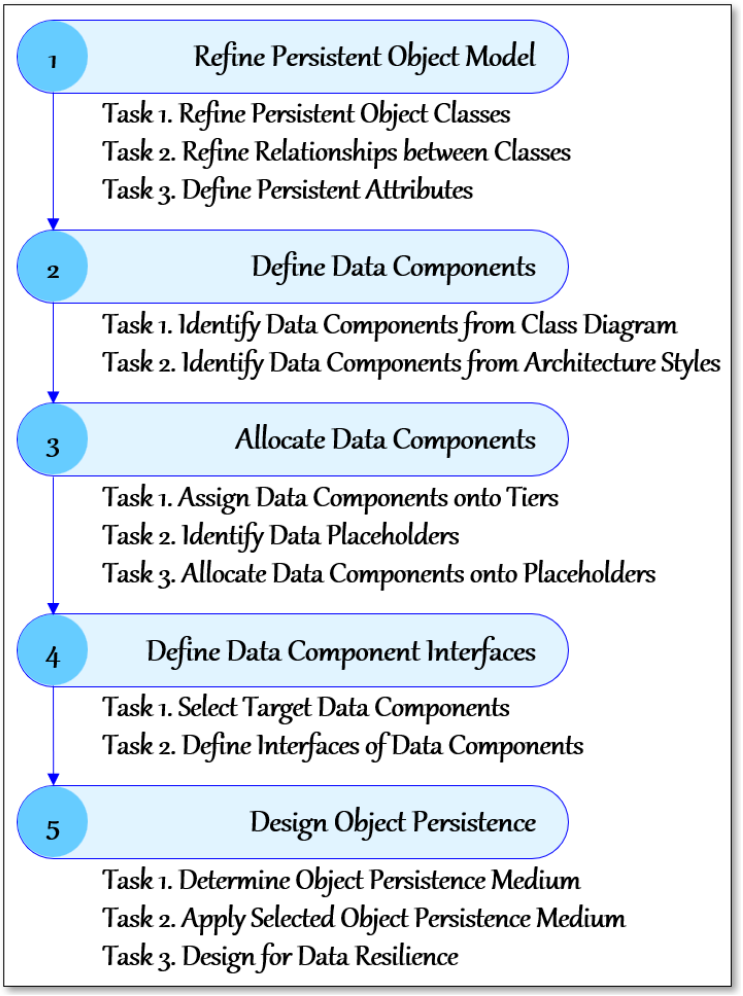
Step 5. Design for Functional Variability

This step is to model the variability inherent in functional components and design them to support flexible configurations that accommodate diverse static and dynamic functional demands, while maintaining system consistency and integrity. It is carried out through the following tasks.

- Task 1, *Identify Variation Points and Variants*, is to identify inherent variability within functional components and explicitly define it through well-specified variation points and their corresponding variants.
- Task 2, *Design Adaptation Schemes for Variability*, is to design appropriate adaptation schemes that effectively manage the identified variability within each functional component.

ACTIVITY A4B. DESIGN FOR INFORMATION VIEW

The purpose of this activity is to design the architecture for the information view of the target system, which involves identifying persistent datasets, defining their relationships, and mapping them to persistent storage. This activity consists of five steps.



✦ Step 1. Refine Persistent Object Model

This step is to produce a well-defined persistent object model by refining and enhancing the context-level object model. It is carried out through the following tasks.

- Task 1, *Refine Persistent Object Classes*, is to refine the persistent object classes defined in the context-level class diagram to ensure they accurately represent the system's data structures.
- Task 2, *Refine Relationships between Classes*, is to refine the relationships among classes, including the specification of cardinalities to indicate instance-level associations.
- Task 3, *Define Persistent Attributes*, is to define the essential persistent attributes for each class in the class diagram, ensuring completeness and alignment with the system's data requirements.

✦ Step 2. Define Data Components

This step is to derive data components from the refined class diagram where a data component is a cohesive unit that consists of closely related classes. It is carried out through the following tasks.

- Task 1, *Identify Data Components from Class Diagram*, is to identify data components by grouping closely related classes based on their structural and semantic relationships.
- Task 2, *Identify Data Components from Architecture Styles*, is to define data components that represent persistent datasets manipulated by functional components according to the selected architecture styles.

✦ Step 3. Allocate Data Components

This step is to assign the data components to their respective data placeholders within a schematic architecture. It is carried out through the following tasks.

- Task 1, *Assign Data Components onto Tiers*, is to allocate each data component to one or more tiers within the schematic architecture, based on its storage and access characteristics.
- Task 2, *Identify Data Placeholders*, is to identify data placeholders represented in the schematic architecture, which serve as designated locations for data component deployment.
- Task 3, *Allocate Data Components onto Placeholders*, is to assign the data components to their corresponding data placeholders, ensuring alignment with the architectural structure and data access requirements.

✦ Step 4. Define Data Component Interfaces

This step is to define the interfaces for selected data components. It is carried out through the following tasks.

- Task 1, *Select Target Data Components*, is to identify the data components that require a precise and well-defined specification of their interfaces.
- Task 2, *Define Interfaces of Data Components*, is to define the interfaces of the selected data components, specifying the operations for accessing and manipulating the underlying persistent data.

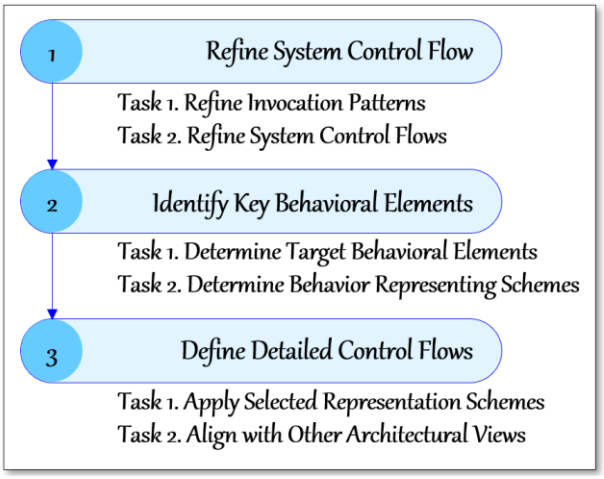
✦ Step 5. Design Object Persistence

This step is to design object persistence for the data components. It is carried out through the following tasks.

- Task 1, *Determine Object Persistence Medium*, is to identify the most appropriate medium for supporting object persistence based on the system's storage and access requirements.
- Task 2, *Apply Selected Object Persistence Medium*, is to apply the chosen persistence medium to provide reliable storage for persistent objects in accordance with the system's design.
- Task 3, *Design for Data Resilience*, is to design the object persistence mechanism to ensure that data remains accessible, consistent, and recoverable in the event of disruptions or failures.

ACTIVITY A4C. DESIGN FOR BEHAVIOR VIEW

The purpose of this activity is to design the architecture for the behavior view of the target system, which focuses on the system's runtime behavior and the detailed control flows of selected behavioral elements. This activity consists of three steps.



Step 1. Refine System Control Flow

This step is to refine the context-level system control flow with detailed observations on the systems' runtime behavior. It is carried out through the following tasks.

- Task 1, *Refine Invocation Patterns*, is to refine the invocation patterns of functional components identified during the behavior context analysis, ensuring they accurately represent component interactions.
- Task 2, *Refine System Control Flows*, is to refine the context-level activity diagrams by incorporating the updated behavioral scope and the refined interaction patterns among components.

Step 2. Identify Key Behavioral Elements

This step is to identify key behavioral elements and to specify the most appropriate behavior representation schemes for these selected elements. It is carried out through the following tasks.

- Task 1, *Determine Target Behavioral Elements*, is to identify the behavioral elements that require detailed control flow modeling based on their complexity, criticality, or interaction characteristics.
- Task 2, *Determine Behavior Representation Schemes*, is to identify the most appropriate representation scheme for modeling the detailed control flow of each selected behavioral element.

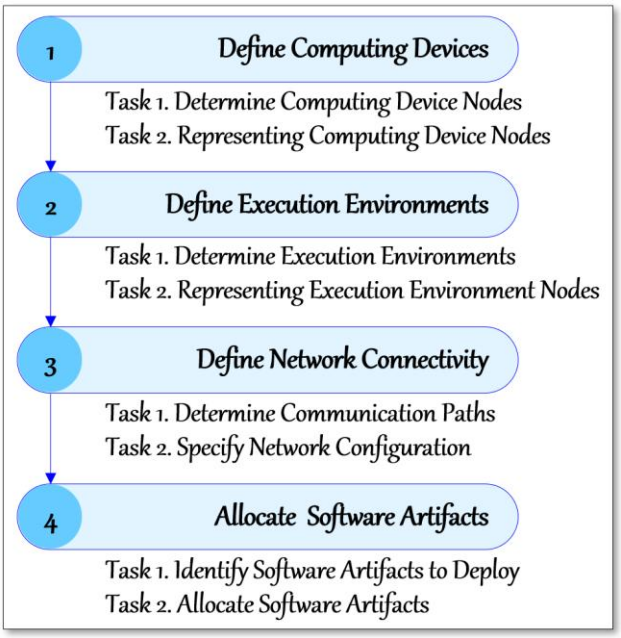
Step 3. Define Detailed Control Flows

This step is to represent the detailed control flows for the selected behavioral elements by adhering to the representation schemes. It is carried out through the following tasks.

- Task 1, *Apply Selected Representation Schemes*, is to define the detailed control flow of each behavioral element using its designated behavior representation scheme.
- Task 2, *Align with Other Architectural Views*, is to analyze the impact of the detailed control flow designs on other architectural views and refine the overall architecture to ensure consistency and alignment across views.

Activity A4d. Design for Deployment View

The purpose of this activity is to design the architecture for the deployment view of the target system, which focuses on the physical and logical configuration of the system's infrastructure. This activity consists of four steps.



✦ Step 1. Define Computing Device Nodes

This step is to specify the hardware configuration of computing device nodes for the target system. It is carried out through the following tasks.

- Task 1, *Determine Computing Device Nodes*, is to identify the computing device nodes required by the system through analysis of its schematic architecture.
- Task 2, *Represent Computing Device Nodes*, is to visually represent the identified computing device nodes using UML notation, applying the stereotype «device» to denote their role in the system.

✦ Step 2. Define Execution Environments

This step is to specify the execution environment of each computing device node in the system. It is carried out through the following tasks.

- Task 1, *Determine Execution Environments*, is to identify the execution environments required for each computing device node, based on the system's deployment requirements.
- Task 2, *Represent Execution Environment Nodes*, is to define and visually represent the execution environment nodes in the deployment diagram, applying the UML stereotype «executionEnvironment».

✦ Step 3. Define Network Connectivity

This step is to define the network connectivity among nodes, represented by communication paths. It is carried out through the following tasks.

- Task 1, *Determine Communication Paths*, is to identify the communication paths among nodes in the deployment diagram, reflecting the system's required interactions across distributed components.
- Task 2, *Specify Network Configuration*, is to define the network configurations for the identified communication paths, including parameters such as protocols, port numbers, bandwidth, and latency.

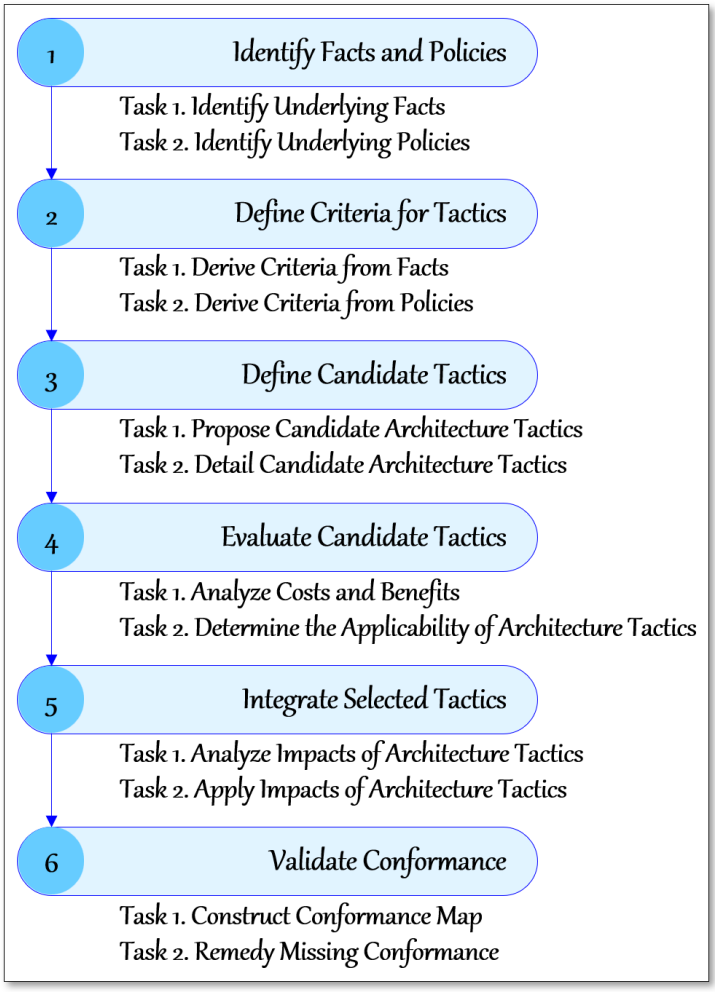
✦ Step 4. Allocate Software Artifacts

This step is to allocate software artifacts onto the identified execution environment nodes. It is carried out through the following tasks.

- Task 1, *Identify Software Artifacts to Deploy*, is to determine the software artifacts that need to be deployed onto execution environment nodes based on the system's architectural design.
- Task 2, *Allocate Software Artifacts*, is to assign the identified software artifacts to the appropriate execution environment nodes within the deployment diagram.

ACTIVITY A5. DESIGN FOR NON-FUNCTIONAL REQUIREMENTS

The purpose of this activity is to design the architecture to address the non-functional requirements of the target system by augmenting the view-based architectural design with architectural tactics that effectively fulfill the identified non-functional requirements. This activity consists of six steps.



✦ Step 1. Identify Facts and Policies

This step is to identify facts and policies that underline the non-functional requirements, serving as the foundational elements for designing architectural tactics. It is carried out through the following tasks.

- Task 1, *Identify Underlying Facts*, is to identify the system-specific facts that are associated with the given non-functional requirements and may influence architectural decisions.

- Task 2, *Identify Underlying Policies*, is to identify the organizational or regulatory policies related to the given non-functional requirements that must be considered in the architectural design.

✦ Step 2. Define Criteria for Tactics

This step is to define the criteria for deriving architectural tactics that address the given non-functional requirement. It is carried out through the following tasks.

- Task 1, *Derive Criteria from Facts*, is to systematically derive criteria from underlying facts by rephrasing them into clear, instructive, and directive statements that support the formulation of effective architecture tactics.
- Task 2, *Derive Criteria from Policies*, is to derive criteria from underlying policies by translating them into clear, instructive, and actionable statements that guide the development of architecture tactics aligned with architectural objectives.

✦ Step 3. Define Candidate Tactics

This step is to define candidate architecture tactics based on the identified criteria. It is carried out through the following tasks.

- Task 1, *Propose Candidate Architecture Tactics*, is to propose candidate architectural tactics derived from the previously identified criteria, ensuring alignment with the targeted non-functional requirements.
- Task 2, *Detail Candidate Architecture Tactics*, is to comprehensively define each candidate architectural tactic by specifying its structure, behavior, and technical implementation details.

✦ Step 4. Evaluate Candidate Tactics

This step is to evaluate the candidate architectural tactics and select the most appropriate ones to be incorporated into the architectural design process. It is carried out through the following tasks.

- Task 1, *Analyze Costs and Benefits*, is to assess the costs and benefits associated with each candidate architectural tactic, considering factors such as implementation effort, resource consumption, and impact on system quality.
- Task 2, *Determine the Applicability of Architecture Tactics*, is to evaluate the suitability of each candidate architectural tactic by determining whether it should be adopted or discarded based on the results of the cost-benefit analysis.

✦ Step 5. Integrate Selected Tactics

This step is to evaluate the impacts of the selected architectural tactics and integrate them into the view-based architectural design. It is carried out through the following tasks.

- Task 1, *Analyze Impacts of Architecture Tactics*, is to evaluate the influence of each selected architectural tactic on the various architectural views and, where applicable, on the schematic architecture.
- Task 2, *Apply Impacts of Architecture Tactics*, is to incorporate the selected architectural tactics into the existing architectural design, ensuring consistency with the outcomes of the preceding impact analysis.

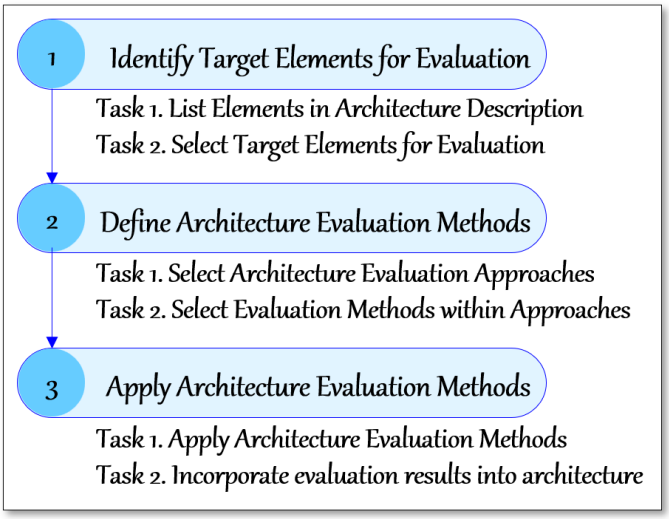
✦ Step 6. Validate Conformance

This step is to validate the conformance of intermediate artifacts to their preceding artifacts using a conformance map. It is carried out through the following tasks.

- Task 1, *Construct Conformance Map*, is to create a conformance map that illustrates the relationships between the artifacts produced in the current step and those generated in the preceding step, highlighting how they align with one another.
- Task 2, *Remedy Missing Conformance*, is to evaluate the conformance relationships represented in the conformance map and address any identified gaps by revising the design to ensure proper alignment and consistency across all related artifacts.

ACTIVITY A6. ARCHITECTURE EVALUATION

The purpose of this activity is to evaluate the architecture design of the target system, with the goal of identifying potential risks, issues, drawbacks, and areas for improvement to ensure alignment with the given functional and non-functional requirements. This activity consists of three steps.



✦ Step 1. Identify Target Elements for Evaluation

This step is to identify the specific elements in the architecture design that require evaluation. It is carried out through the following tasks.

- Task 1, *List Elements in Architecture Description*, is to enumerate all design elements included in the architectural description of the target system.
- Task 2, *Select Target Elements for Evaluation*, is to identify and select specific elements within the architectural description that require focused evaluation based on their importance, complexity, or potential risk.

✦ Step 2. Define Architecture Evaluation Methods

This step is to determine and select appropriate methods for evaluating the design elements of architecture description. It is carried out through the following tasks.

- Task 1, *Select Architecture Evaluation Approaches*, is to choose the most appropriate evaluation approaches for assessing the selected design elements, considering the nature and criticality of each element.

- Task 2, *Select Evaluation Methods within Approaches*, is to identify the most suitable evaluation methods for each chosen architecture evaluation approach, ensuring that they are well-aligned with the characteristics of the selected design elements.

✦ Step 3. Apply Architecture Evaluation Methods

This task is to apply selected architecture evaluation methods and enhance the architecture design by incorporating feedback from the evaluation results. It is carried out through the following tasks.

- Task 1, *Evaluate Architecture Design with Selected Methods*, is to assess the architecture design by systematically applying the selected evaluation methods to the targeted design elements.
- Task 2, *Incorporate Evaluation Results into Architecture*, is to refine the architecture design by systematically integrating feedback and recommendations derived from the evaluation results.

UAP Resources

The following resources support effective learning and mastery of the UAP methodology.

- **Book**: S. Kim and M. Kim, *Hands-on Software Architecture: Unified Architecture Process*, Springer, 2025.

  This book provides comprehensive details of the UAP process model, design guidelines, validation checklists, and architecture styles.

- **Companion Website**: www.handsonse.org

  This website provides a rich set of downloadable resources, including the UAP overview, charts, guides, architecture description (AD) templates, and case studies.

- **Training Programs**

  A curated set of UAP training programs is available to deliver the right technical expertise for diverse needs. Representative programs include *UAP Foundations, UAP Intermediate, and UAP Advanced*.