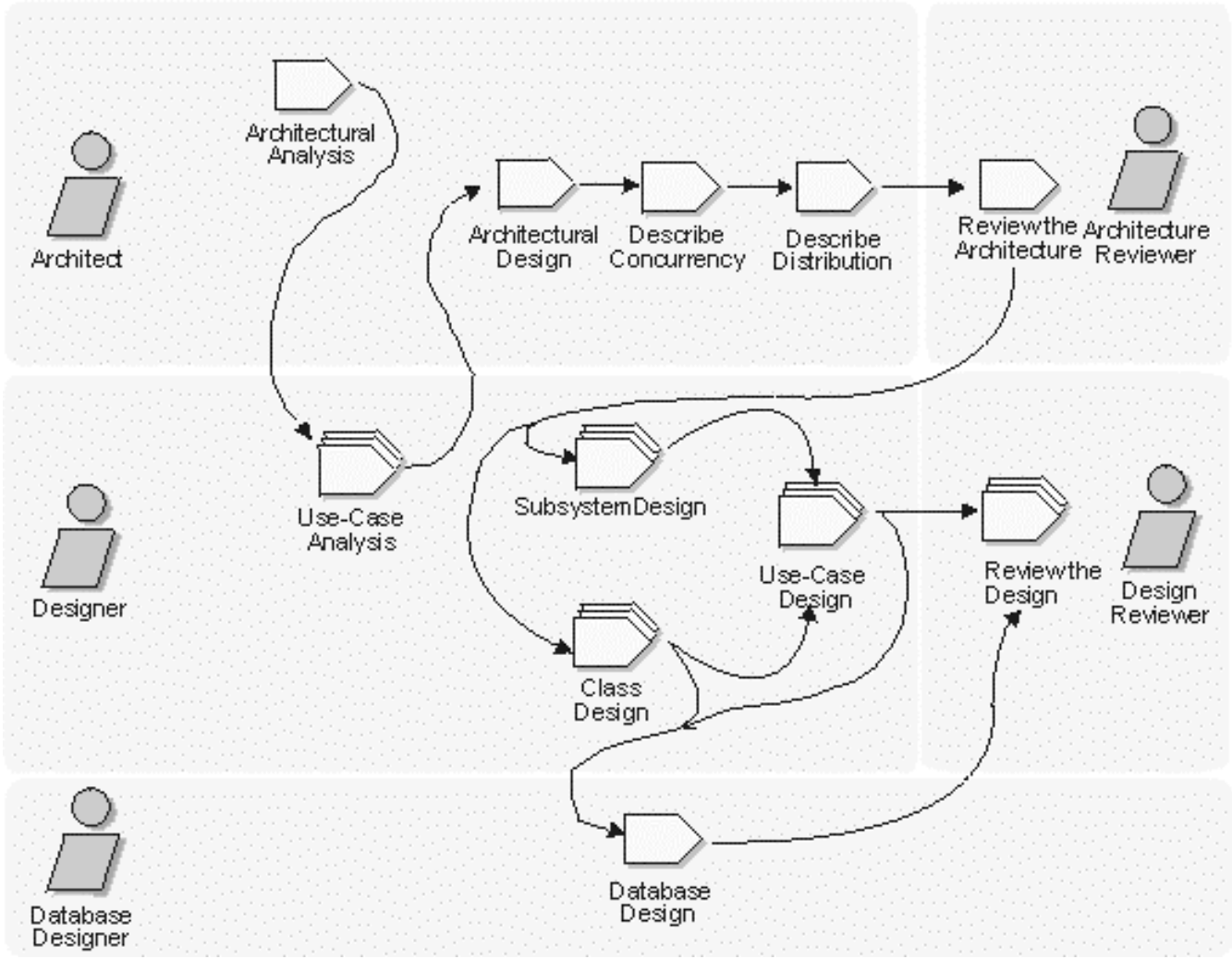
Analysis and Design

Mar 12, 2000 | v1.0

Goals

- To transform the requirements into a design of the system to-be
- To evolve a robust architecture for the system
- To adapt the design to match the implementation environment, designing it for performance

Workflow



Architectural Analysis

- Define Modeling Conventions
  - → to ensure that the representation of the architecture and design are consistent across teams and iterations; design guidelines
- Define the High-Level Organization of Subsystems
  - → to create an initial structure for the Design Model
- Identify Analysis Mechanisms
  - → to define the architectural patterns and services used by designers
- Identify Key Concepts
  - → to identify the key abstractions (representation of concepts identified during business modeling and requirement activities) that the system must handle; initial classes
- Create Use-Case Realizations
  - → to create the Design Model artifacts used to express the behavior of the use cases
- Review the Results
  - → to ensure that the results of architectural analysis is complete and consistent

Use-Case Analysis

- Supplement the Descriptions of the Use Case
  - → to capture additional information needed in order to understand the required internal behavior of the system that may be missing from the use-case description written for the customer of the system; ‘white box’ description

For each use case realization

- Find Classes from Use-Case Behavior
  - → to identify a candidate set of analysis classes capable of performing the behavior described in use cases
- Distribute Use-Case Behavior to Classes
  - → to express the use-case behavior in terms of collaborating analysis classes and to determine the responsibilities of analysis classes

Use-Case Analysis (cont.)

For each resulting analysis class

- Describe Responsibilities
  - → to describe the responsibilities of a class of objects identified from use-case behavior
- Describe Attributes and Associations
  - → to define attributes, to establish aggregations and associations between analysis classes and to describe event dependencies between analysis classes
- Qualify Analysis Mechanisms
  - → to identify analysis mechanisms (if any) used by the class and to provide additional information about how the class applies the analysis mechanism
- Unify Analysis Classes
  - → to ensure that each analysis class represents a single well-defined concept, with non-overlapping responsibilities
- Evaluate Your Results
  - → to verify that the analysis objects are consistent and meet the functional requirements

Architectural Design

- Identify Design Mechanisms
  - → to categorize clients of analysis mechanisms, to invent the implementation mechanisms, to map design mechanisms to implementation mechanisms and to document architectural mechanisms
- Identify Design Classes and Subsystems
  - → to refine the analysis classes, categorizing them as design classes or subsystems
- Identify Interfaces
  - → to identify the interfaces of the subsystems based on their responsibilities
- Identify Reuse Opportunities
  - → to identify where existing subsystems and/or components may be reused based on their interfaces

Architectural Design (cont.)

- Reverse-engineer components and databases
  - → to incorporate potentially reusable model elements from other projects, external sources or prior iterations
- Define the Low-level Organization of Subsystems
  - → to organize the lower layers of the Design Model
- Include Architecturally Significant Model Elements in the Logical View
  - → to document the results of Architectural Design

Describe Concurrency

- Define Concurrency Requirements
  - → to define the extent to which parallel execution of tasks is required for the system
- Identify Processes
  - → to define the processes and threads which will exist in the system
- Identify Process Lifecycles
  - → to identify when processes and threads are created and destroyed
- Identify Inter-Process Communication Mechanisms
  - → to identify the mean by which processes and threads will communicate
- Allocate Inter-Process Coordination Resources
  - → to allocate scarce resources and to manage potential performance bottlenecks
- Map Processes onto the Implementation Environment
  - → to map processes onto the concepts supported by the implementation environment
- Distribute Model Elements Among Processes
  - → to determine which processes classes and subsystems should execute within

Describe Distribution

- Define the network configuration
  - → to understand the configuration and topology of the network
- Allocate processes to nodes
  - → to distribute the workload of the system
- Evaluate Your Results

Subsystem Design

- Distribute Subsystem Behavior to Subsystem Elements
  - → to specify the internal behaviors of the subsystem and to identify new classes or subsystems needed to satisfy subsystem behavioral requirements
- Document Subsystem Elements
  - → to document the internal structure of the subsystem
- Describe Subsystem Dependencies
  - → to document the interfaces upon which the subsystem is dependent

Class Design

- Create Initial Design Classes
  - → to design boundary, entity and control classes
- Identify Persistent Classes
  - → to identify classes that need to be persistently stored, e.g. treated by database
- Define Class Visibility
  - → for each class to determine the class visibility within the package in which it resides
- Define Operations
  - → to identify, name and describe the operations, define operation visibility and define class operations
- Define Methods
  - → to decide how operations are to be implemented (e.g. how parameters are to be implemented and how any special algorithms to be used, etc.)
- Define States
  - → to describe the object states for some classes and operations

Class Design (cont.)

- Define Attributes
  - → to identify attributes needed by the class to carry out its operations
- Define Dependencies
  - → to define dependencies between communicating
- Define Associations
  - → to specify the relationships between classes those of which instances communicate
- Define Generalizations
  - → to organize classes into a generalization hierarchy to reflect common behavior and common structure
- Handle Non-Functional Requirements in General
  - → to refine the design classes in order to handle non-functional requirements (e.g. performance, re-usability of existing components, programming language constraints, security, distribution, etc.)
- Evaluate Your Results

Use-Case Design

- Describe Interactions Between Design Objects
  - → for each use-case realization to illustrate the interactions between its participating design objects
- Simplify Sequence Diagrams using Subsystems (optional)
  - → to simplify the sequence diagrams by replacing their large subsections with a single message to the subsystem
- Describe Persistence-related Behavior
  - → to specify treating of persistent objects, e.g. writing, reading, deleting, modeling of transactions, handling of errors, handling of concurrency control, etc.
- Refine the Flow of Events Description
  - → to add further description into sequence diagrams, e.g. textual descriptions, algorithm, extension points, etc.
- Unify Classes and Subsystems
  - → to unify the identified classes and subsystems in taking into account naming, behavior, consistency, etc.
- Evaluate Your Results

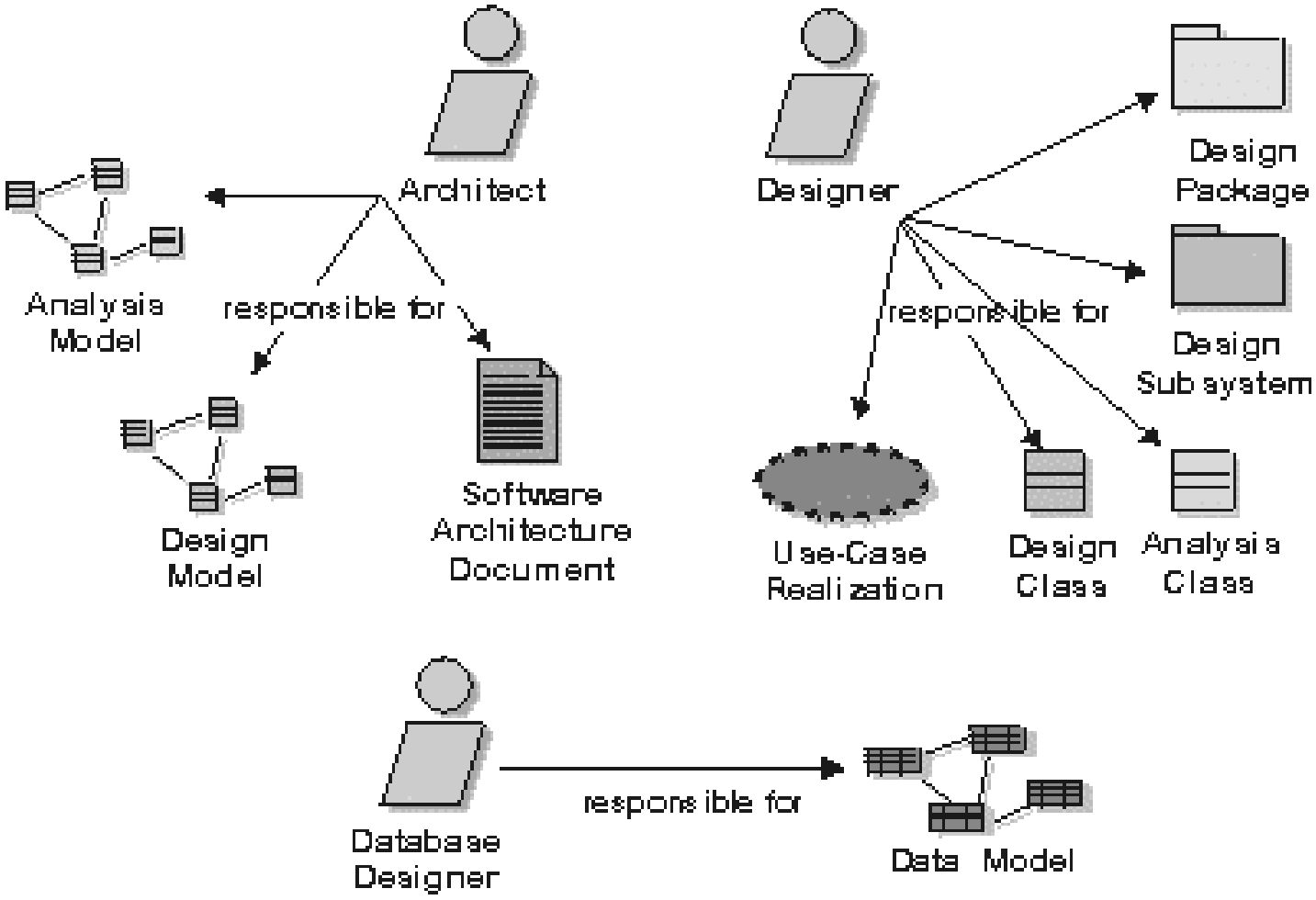
Database Design

- Map Persistent Design Classes to the Data Model
  - → to create define/refine the data model to support storage and retrieval of persistent classes
- Optimize the Data Model for Performance
  - → to optimize the database data structures for performance
- Optimize Data Access
  - → to provide for efficient data access using indexing
- Define Storage Characteristics
  - → to define the space requirements and disk page organization of the database

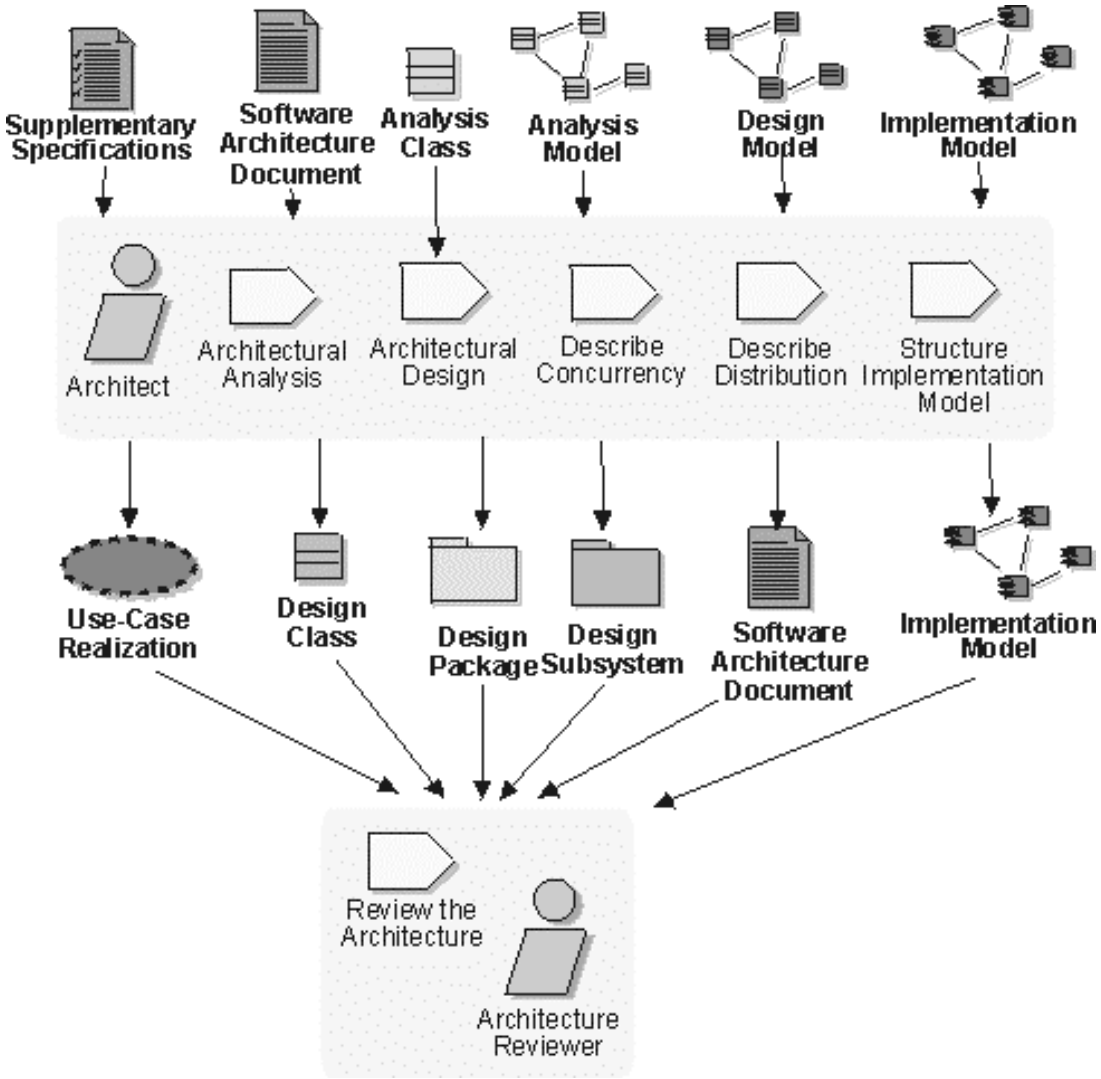
Database Design (cont.)

- Define Reference Tables
  - → to define standard reference tables used across the project and to define default values for data attributes
- Define Data and Referential Integrity Enforcement Rules
  - → to ensure the integrity of the database
- Distribute Class Behavior to the Database
  - → to determine the behavior of the class which can be distributed to, and implemented by, the database
- Review the Results

Workers and Artifacts



Workflow Detail: Define the Architecture



Workflow Detail: Elaborate the Design