6 Identify Design Elements

IBM Software Group

Mastering Object-Oriented Analysis and Design with UML

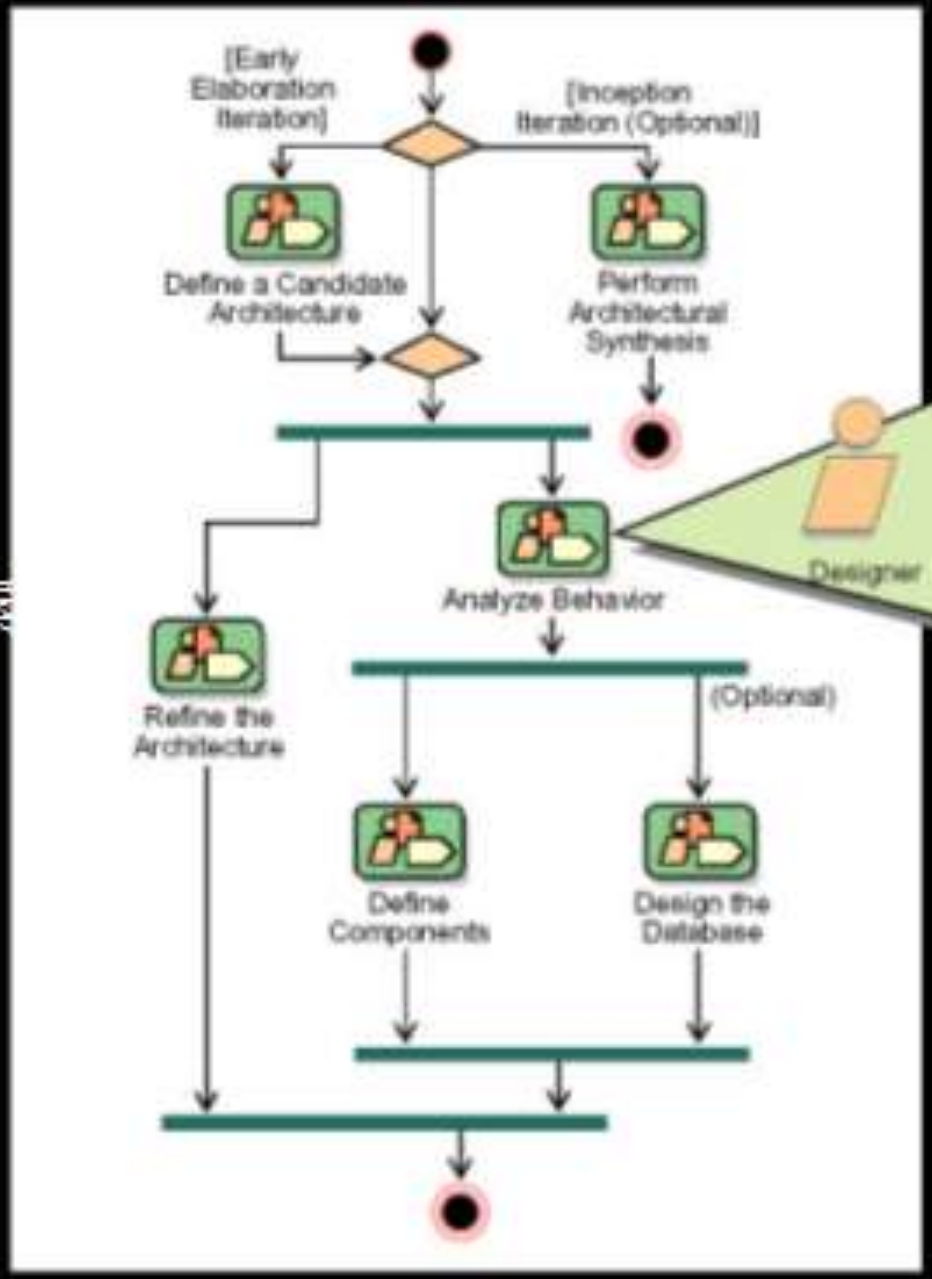
Module 6: Identify Design Elements

Rational. software

架构分析

识别设计元素
运行时架构
描述分布

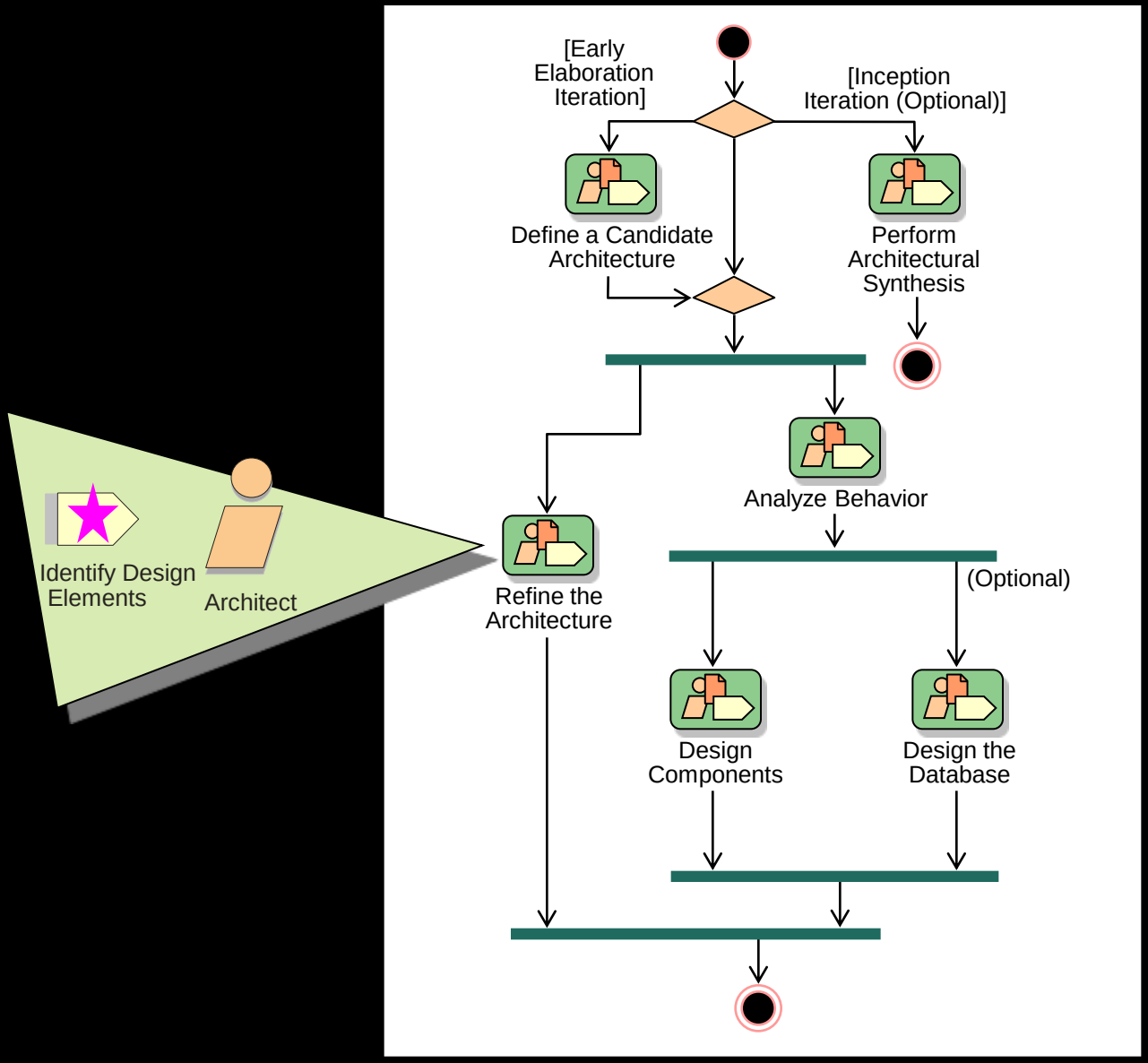
用例设计
子系统设计
类设计

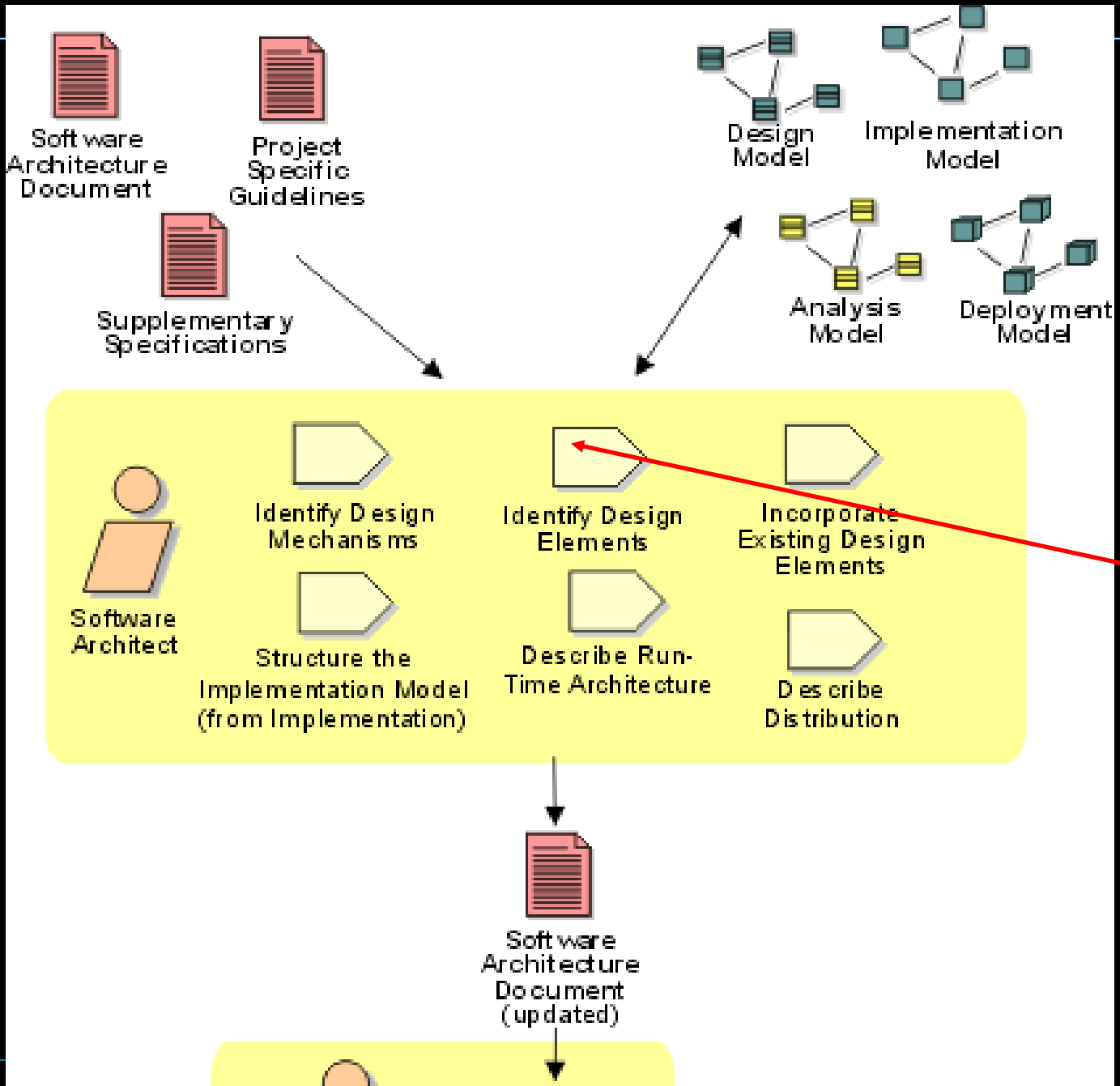


用例分析

数据库设计

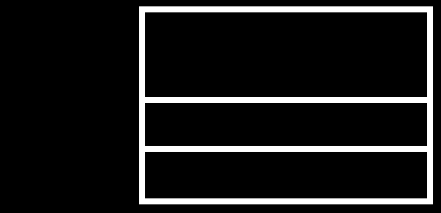
Identify Design Elements in Context



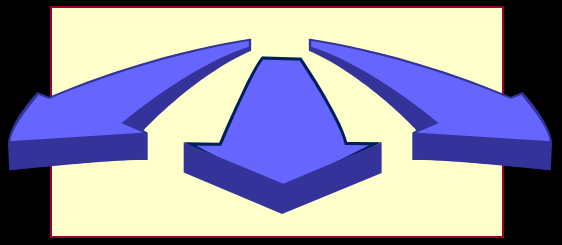

Software Architecture Document
Project Specific Guidelines
Supplementary Specifications
Design Model
Implementation Model
Analysis Model
Deployment Model
Software Architect
Identify Design Mechanisms
Identify Design Elements
Incorporate Existing Design Elements
Structure the Implementation Model (from Implementation)
Describe Run-Time Architecture
Describe Distribution
Software Architecture Document (updated)

Identify Design Elements Steps

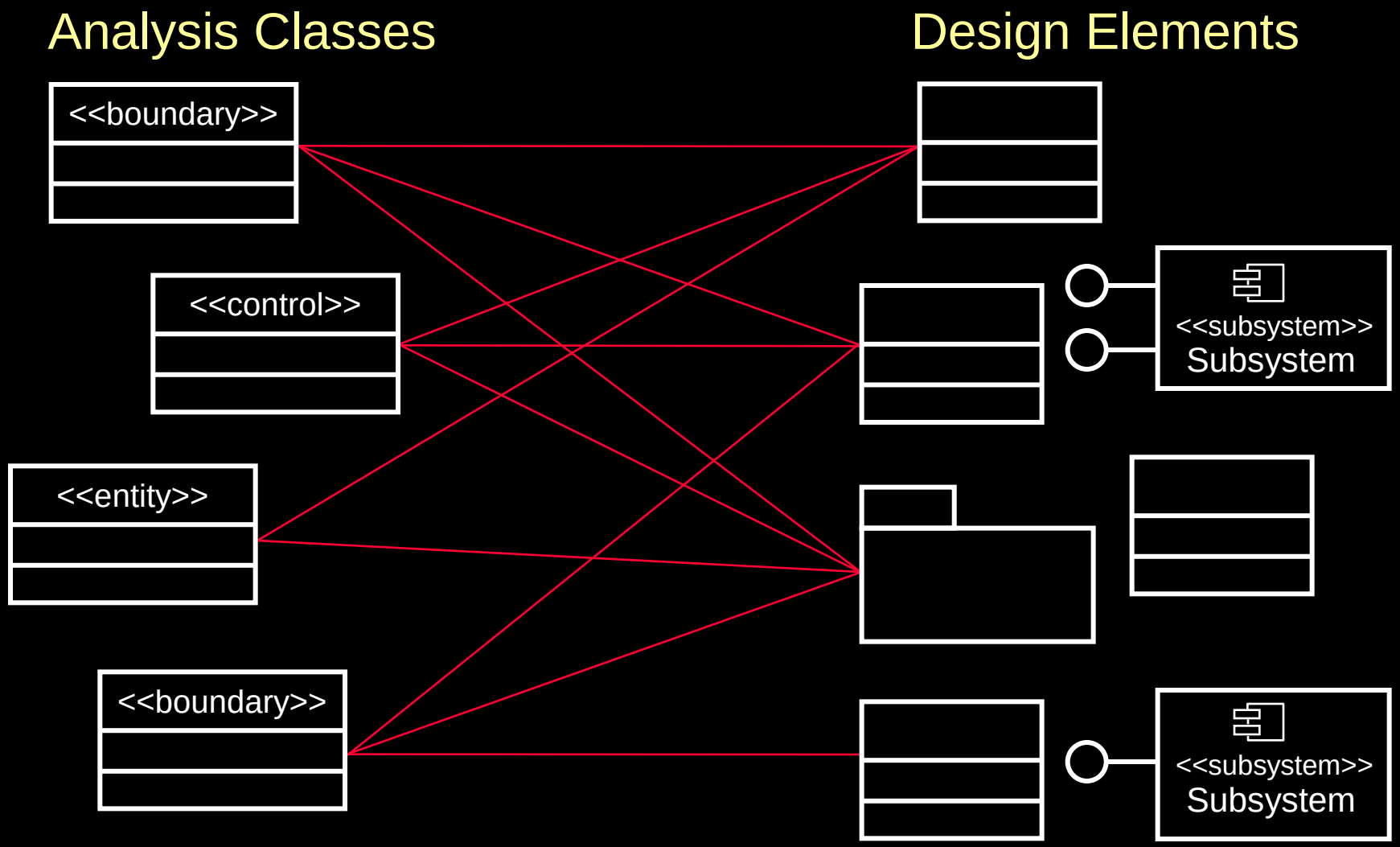
- ★ Identify classes and subsystems
- Identify subsystem interfaces
- Identify reuse opportunities
- Update the organization of the Design Model
- Checkpoints



Analysis Classes

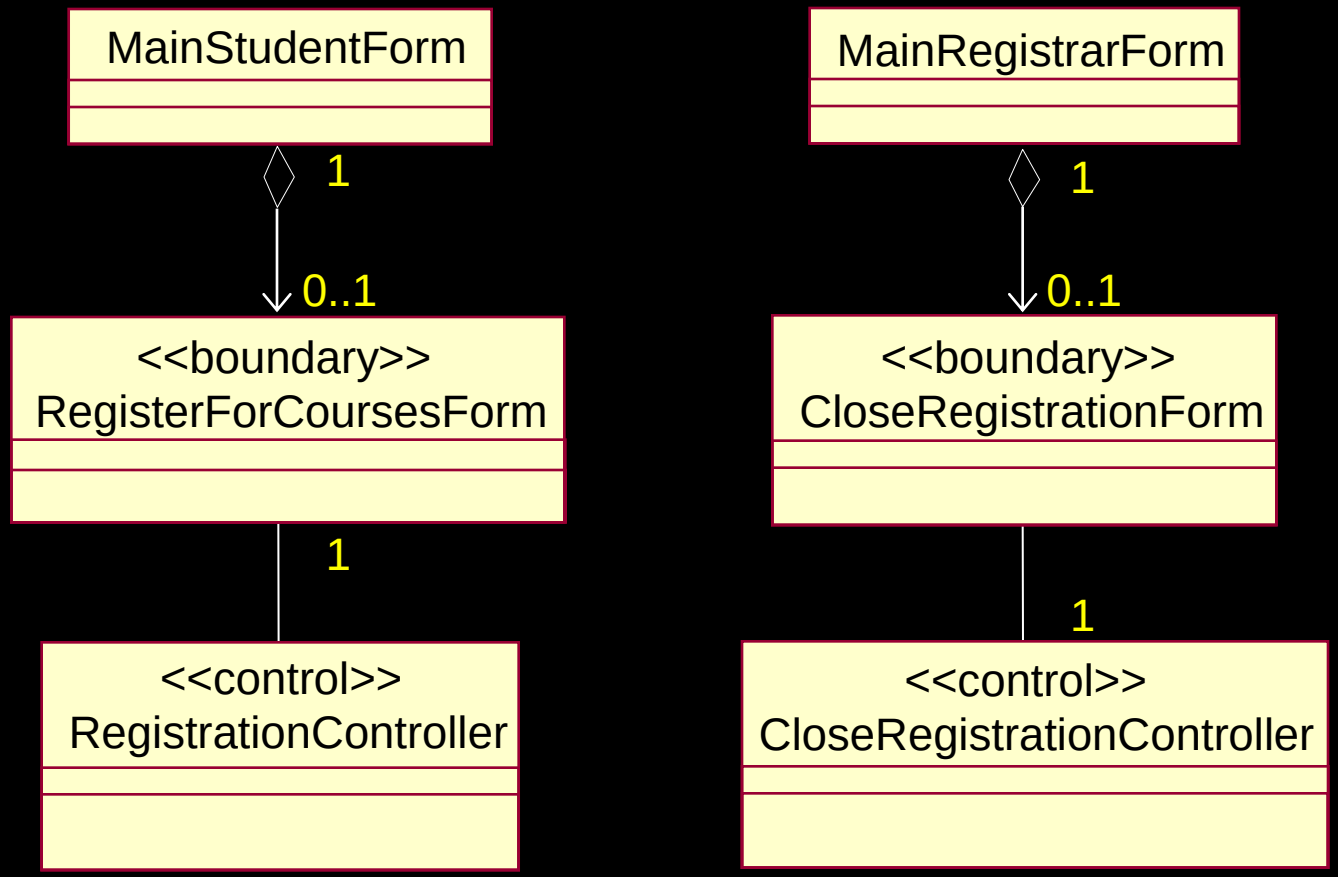


From Analysis Classes to Design Elements

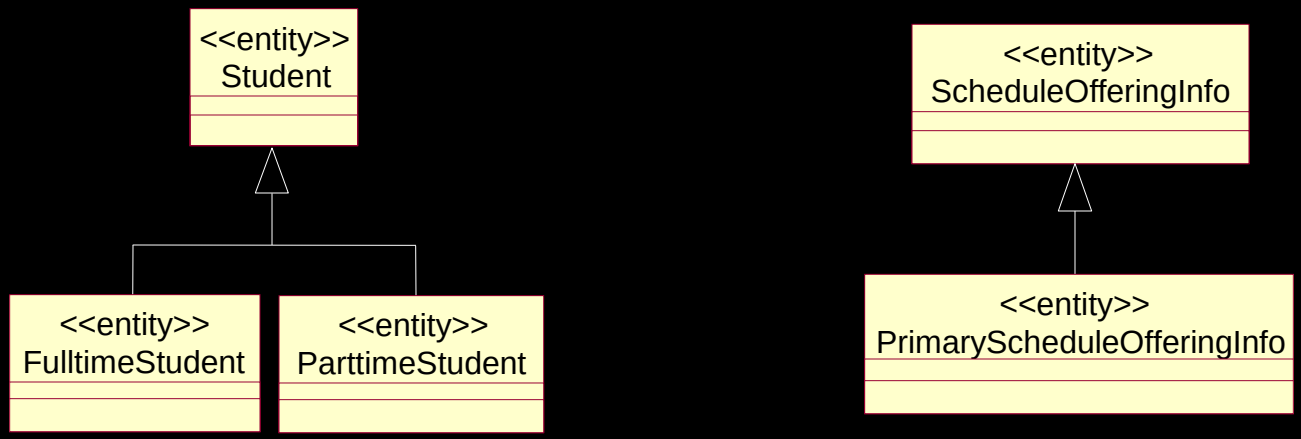


Many-to-Many Mapping

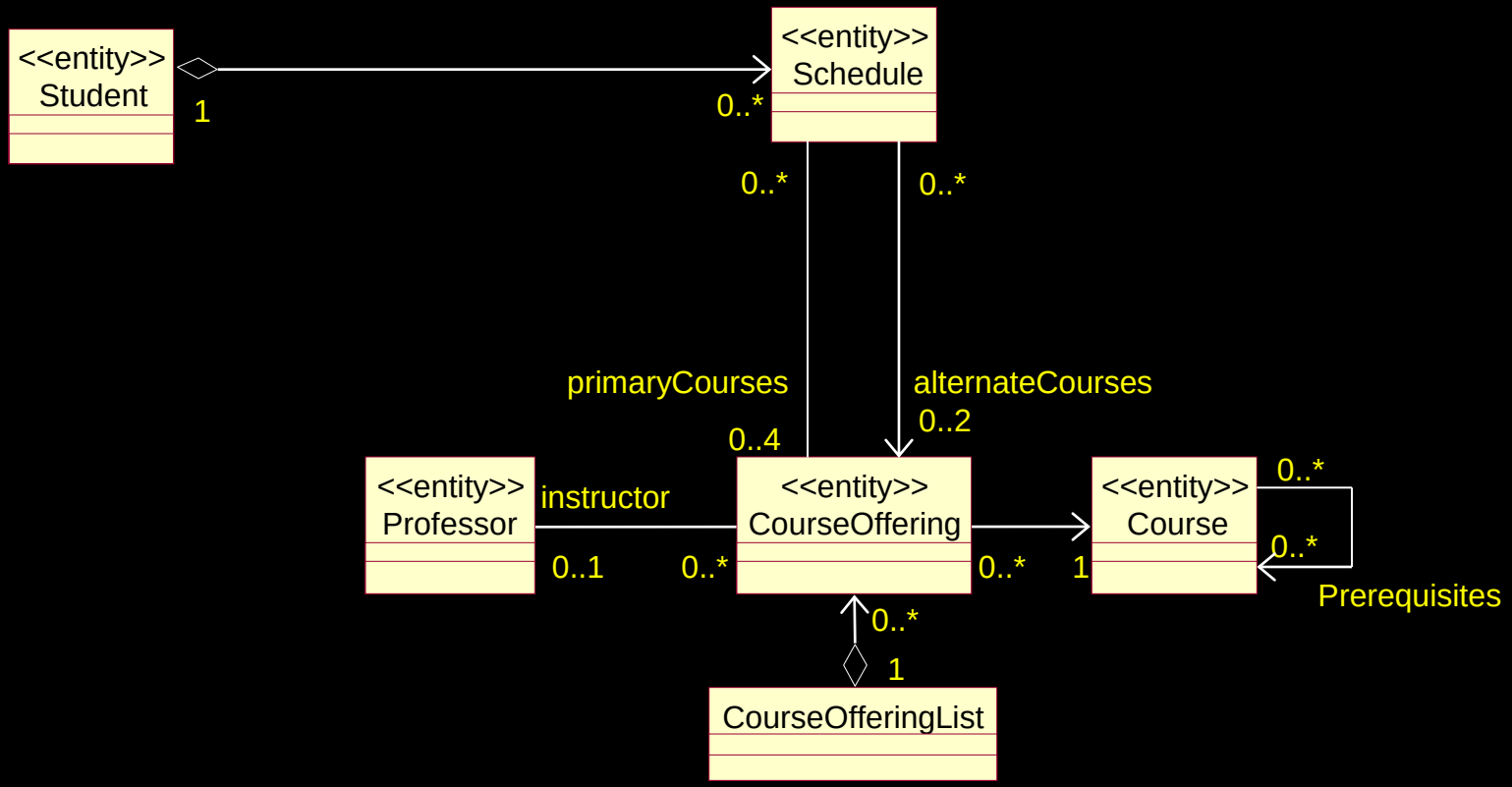
Example: Registration Package



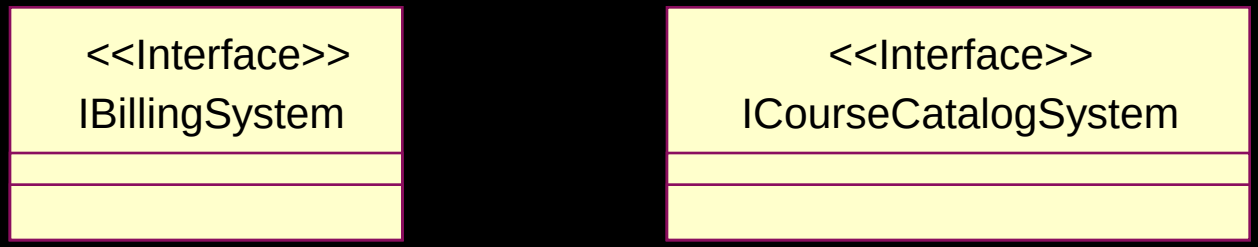
Example: University Artifacts Package: Generalization



Example: University Artifacts Package: Associations

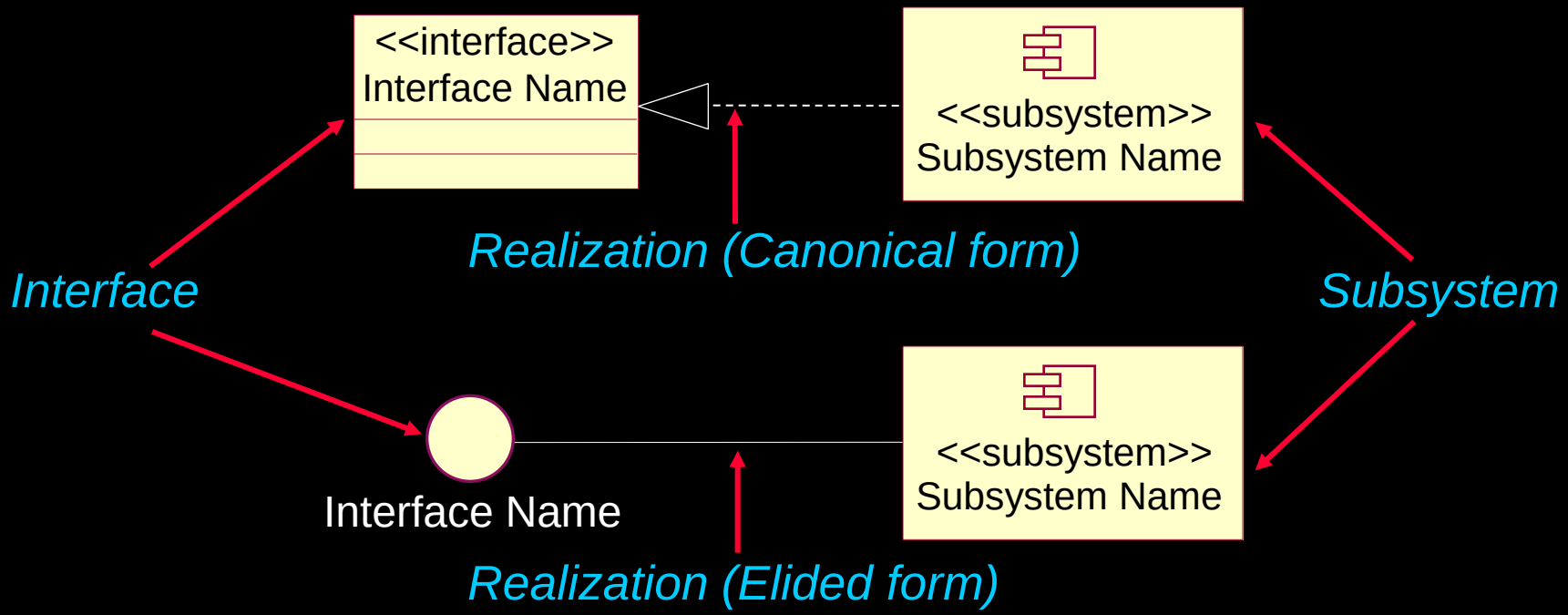


Example: External System Interfaces Package



Review: Subsystems and Interfaces

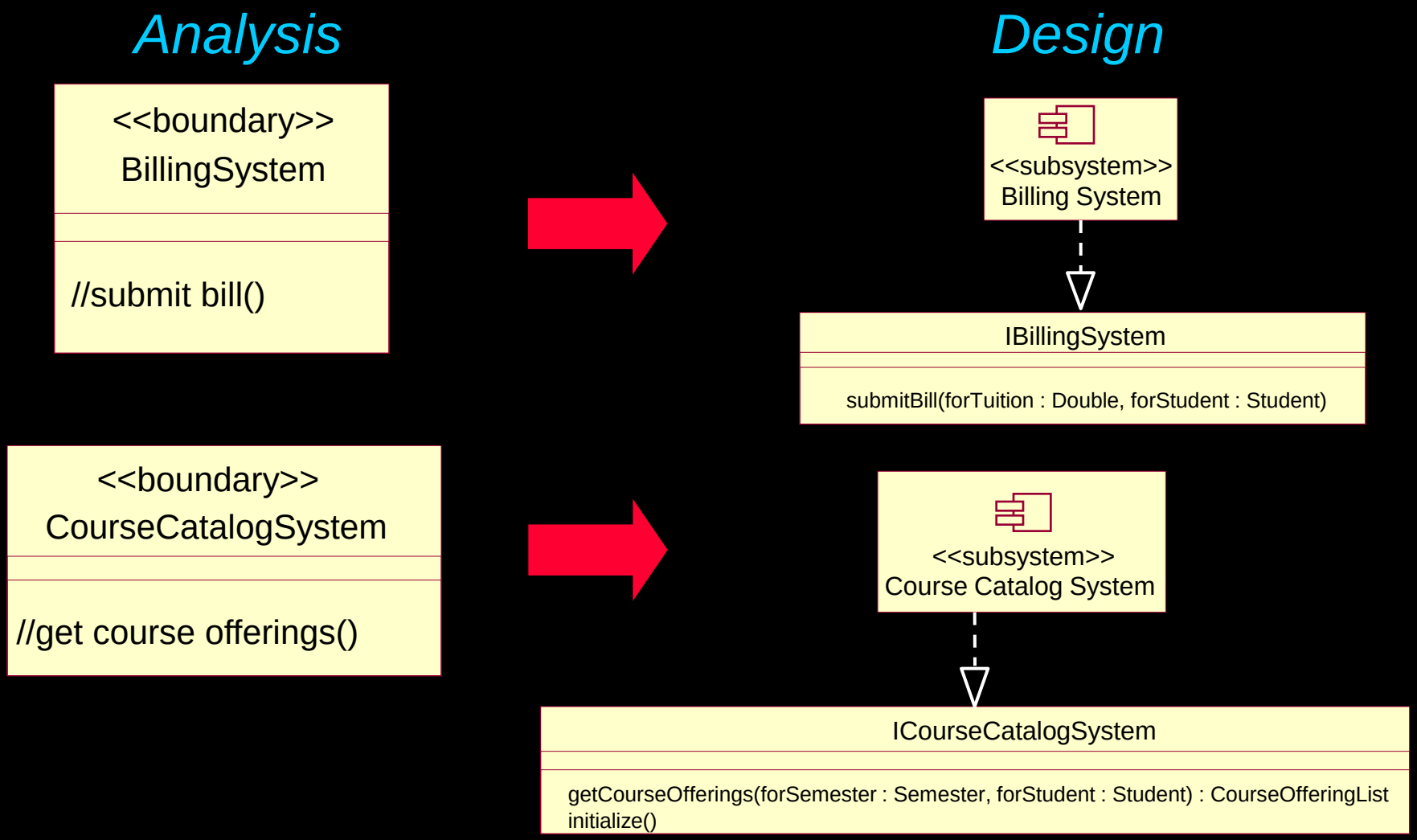
- Realizes one or more interfaces that define its behavior



Identify Design Elements Steps

- Identify classes and subsystems
- ★ Identify subsystem interfaces
- Identify reuse opportunities
- Update the organization of the Design Model
- Checkpoints

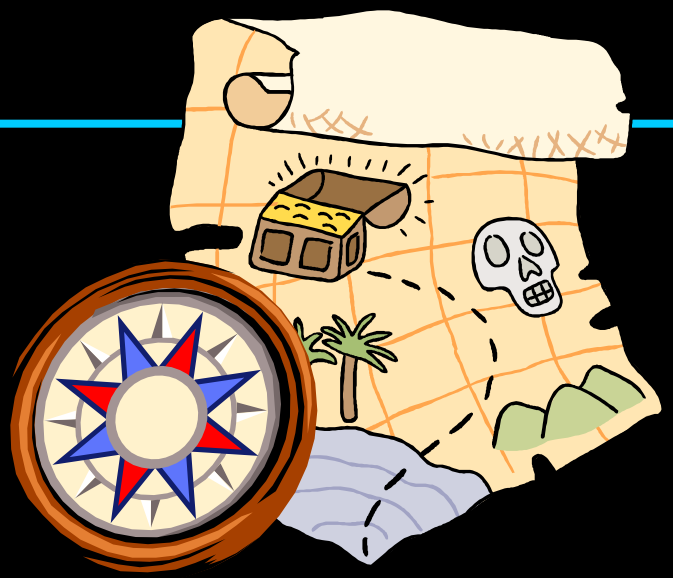
Example: Design Subsystems and Interfaces



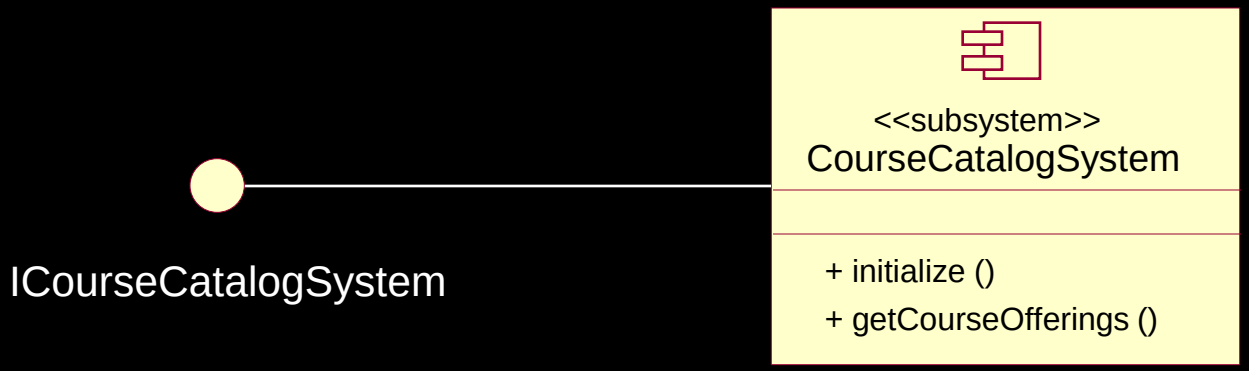
All other analysis classes map directly to design classes.

Example: Analysis-Class-To-Design-Element Map

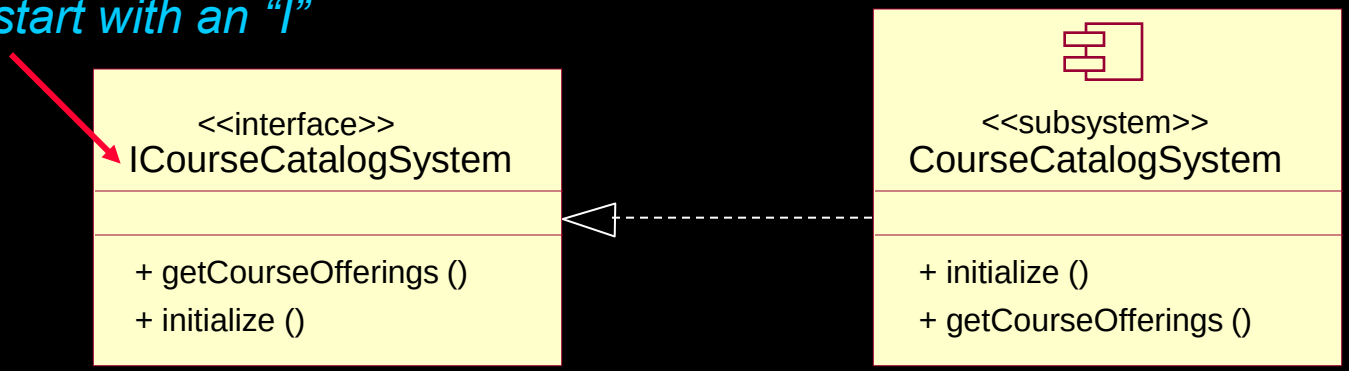
Analysis Class	Design Element
CourseCatalogSystem	CourseCatalogSystem Subsystem
BillingSystem	BillingSystem Subsystem
All other analysis classes map directly to design classes	



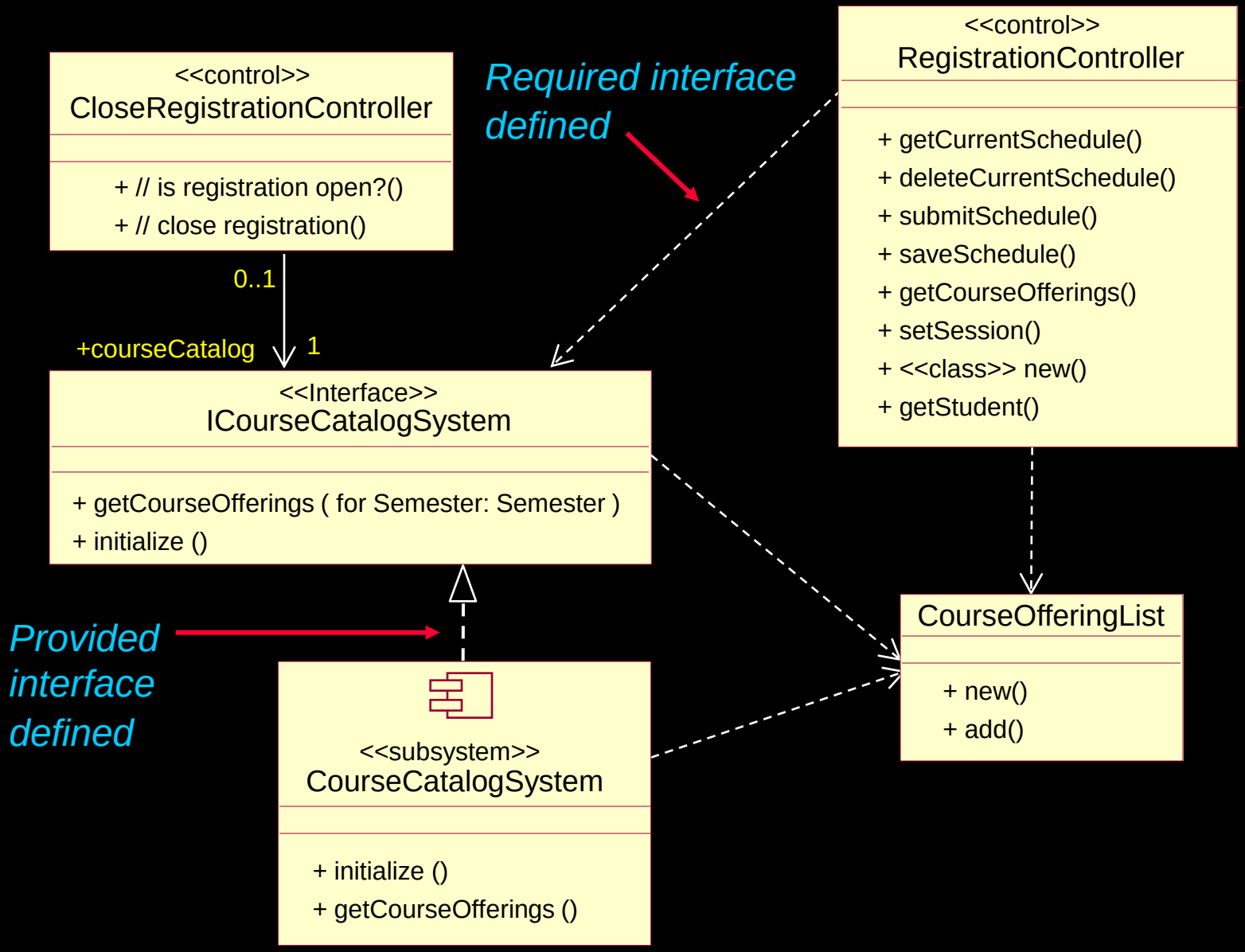
Modeling Convention: Subsystems and Interfaces



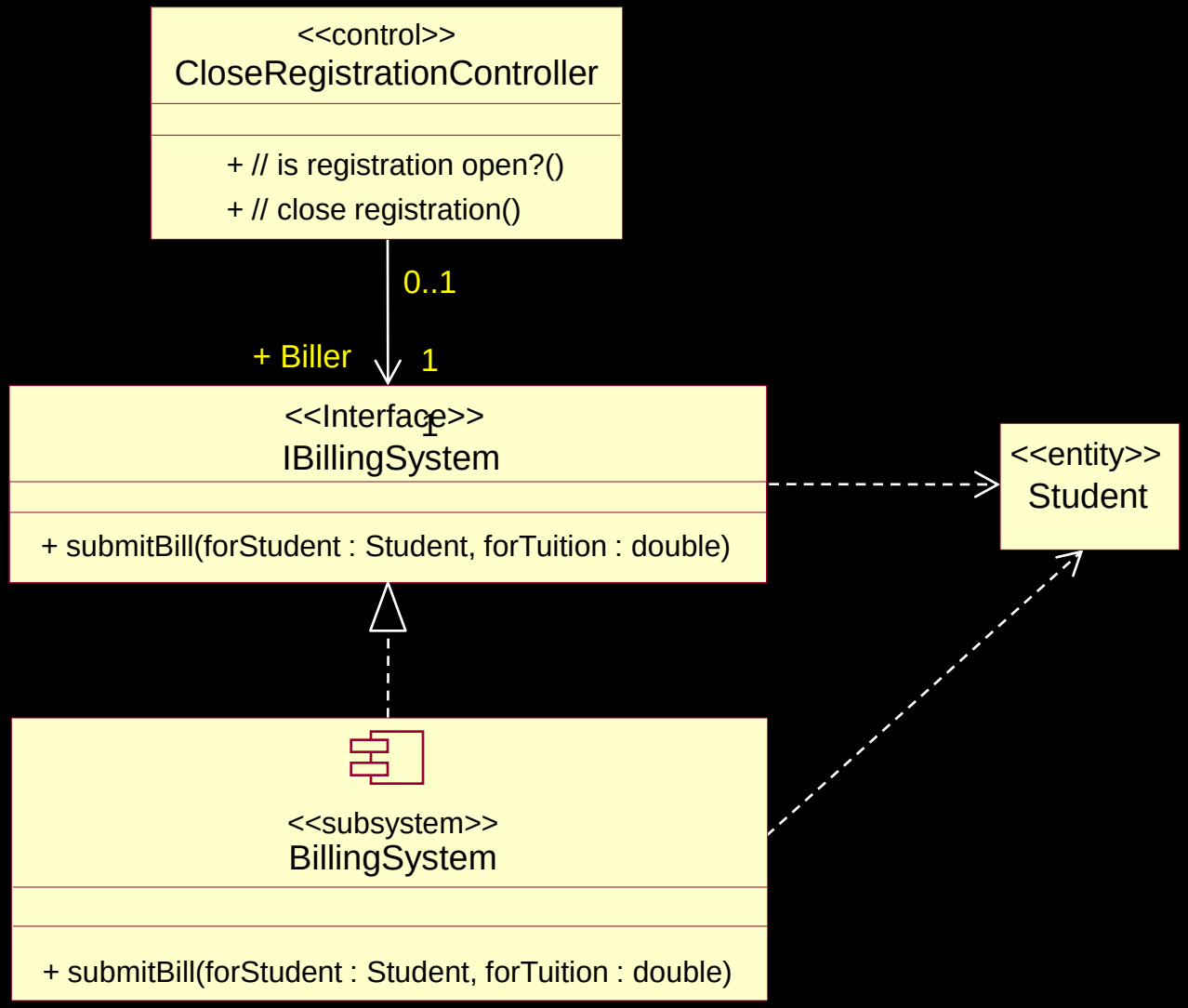
Interfaces start with an "I"



Example: Subsystem Context: CourseCatalogSystem



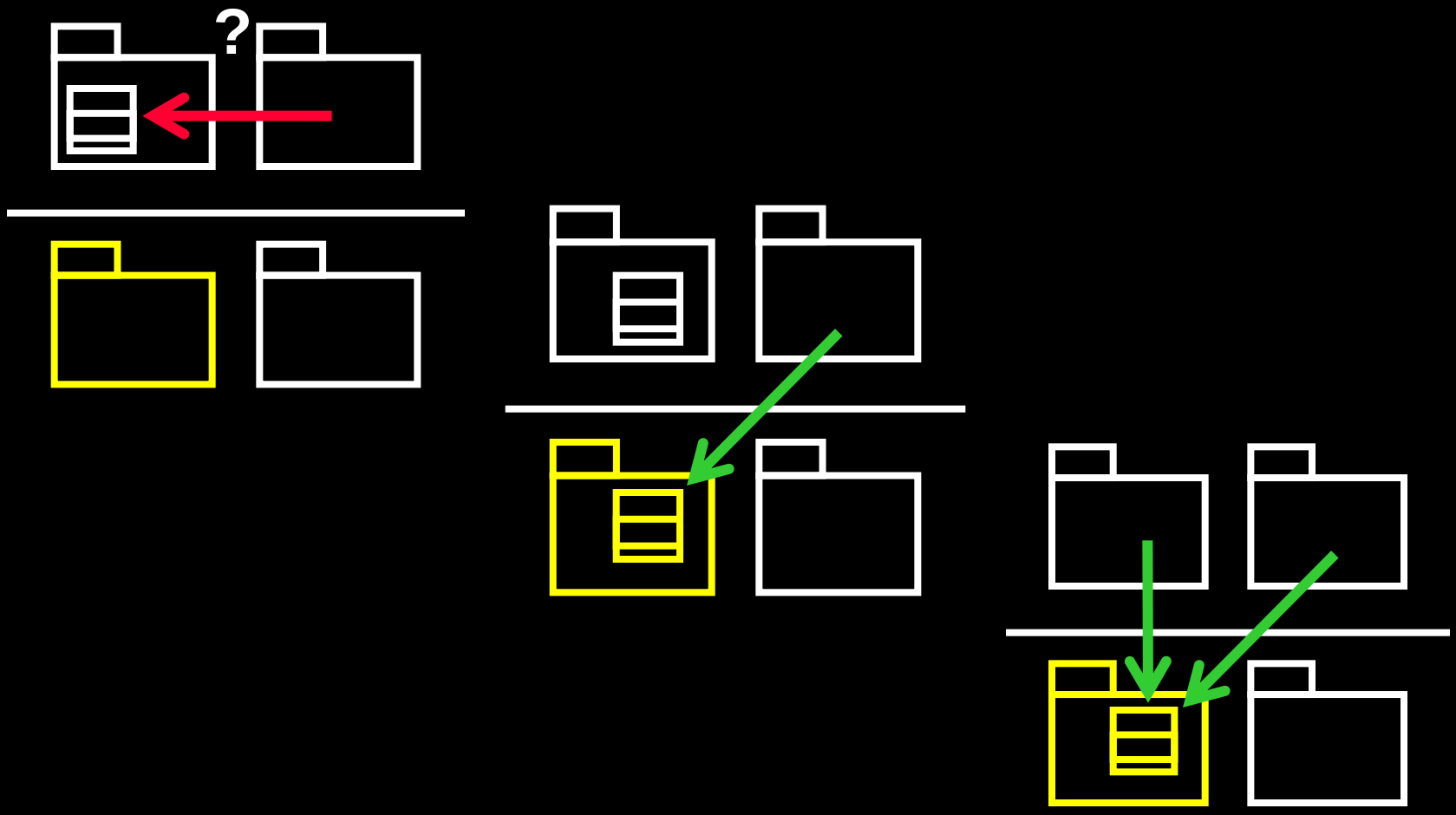
Example: Subsystem Context: Billing System



Identify Design Elements Steps

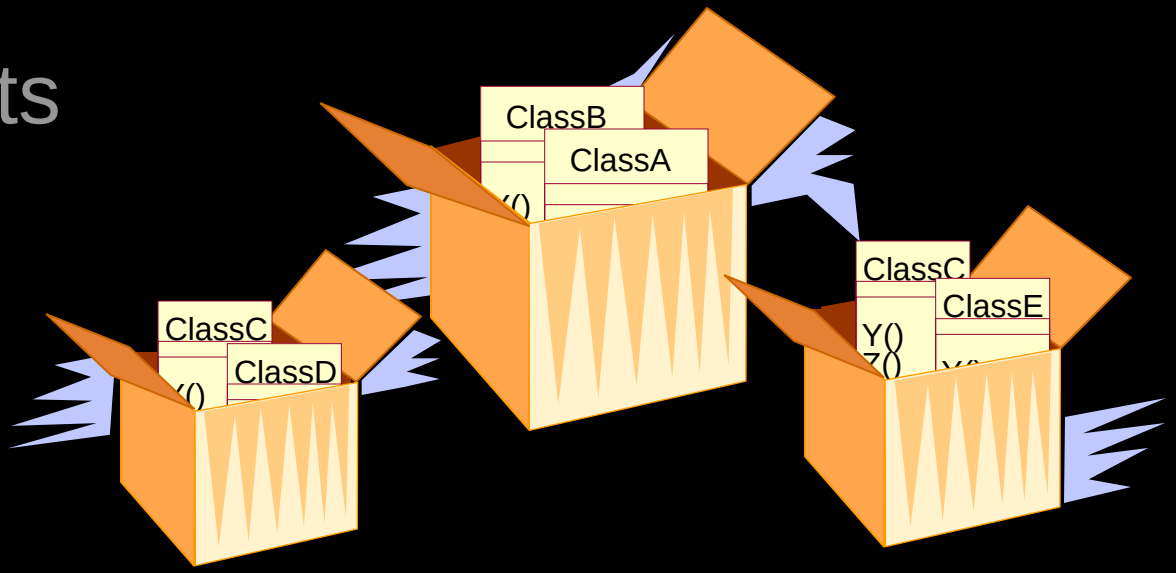
- Identify classes and subsystems
- Identify subsystem interfaces
- ★ Identify reuse opportunities
- Update the organization of the Design Model
- Checkpoints

Reuse Opportunities Internal to System

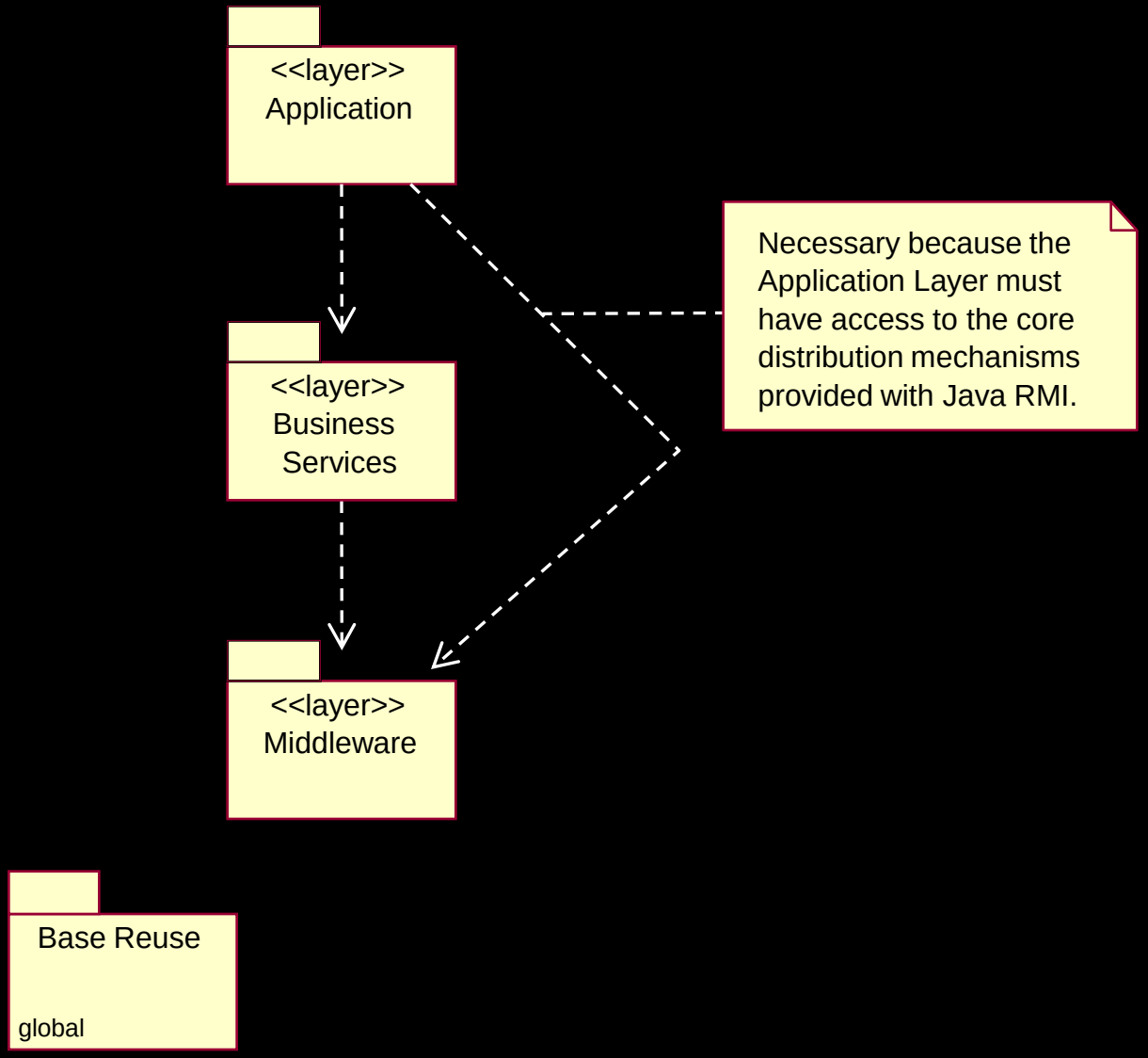


Identify Design Elements Steps

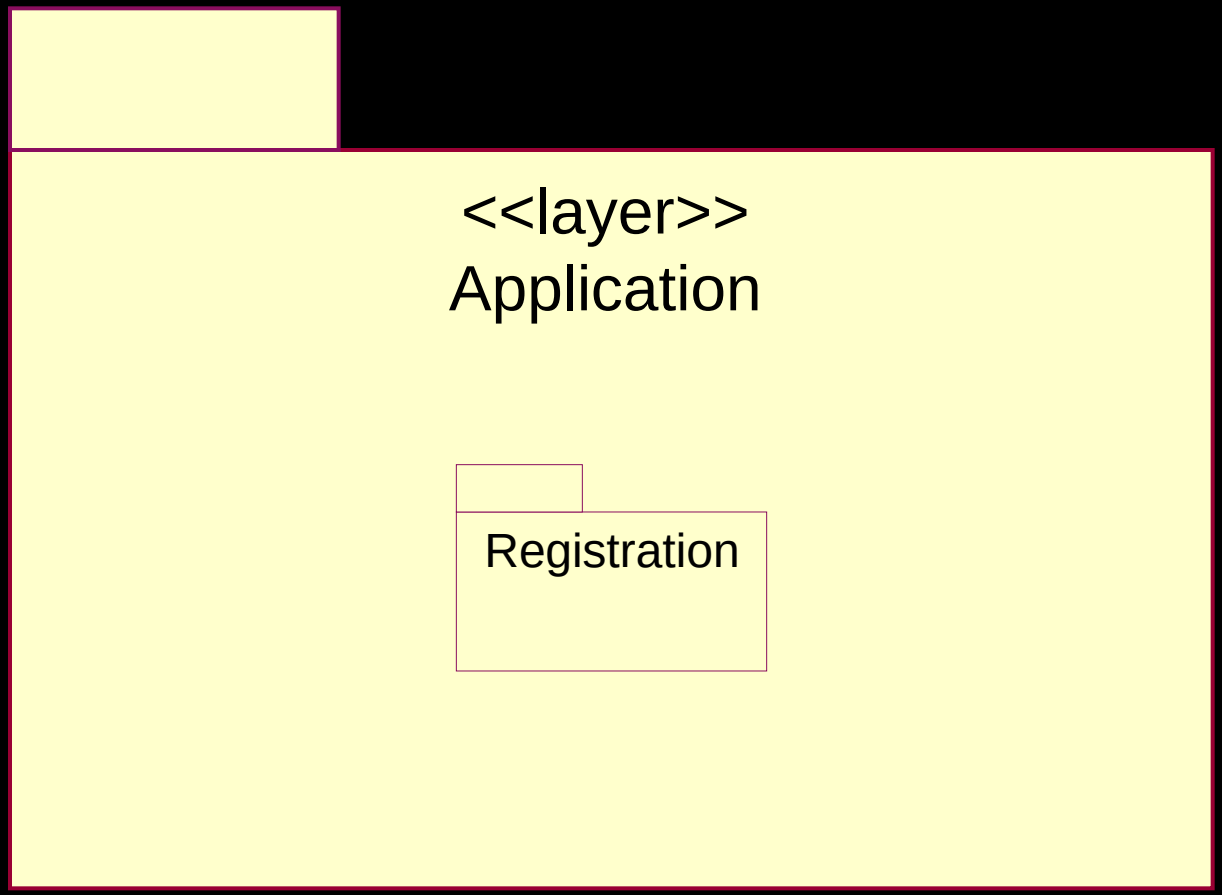
- Identify classes and subsystems
- Identify subsystem interfaces
- Identify reuse opportunities
- ★ Update the organization of the Design Model
- Checkpoints



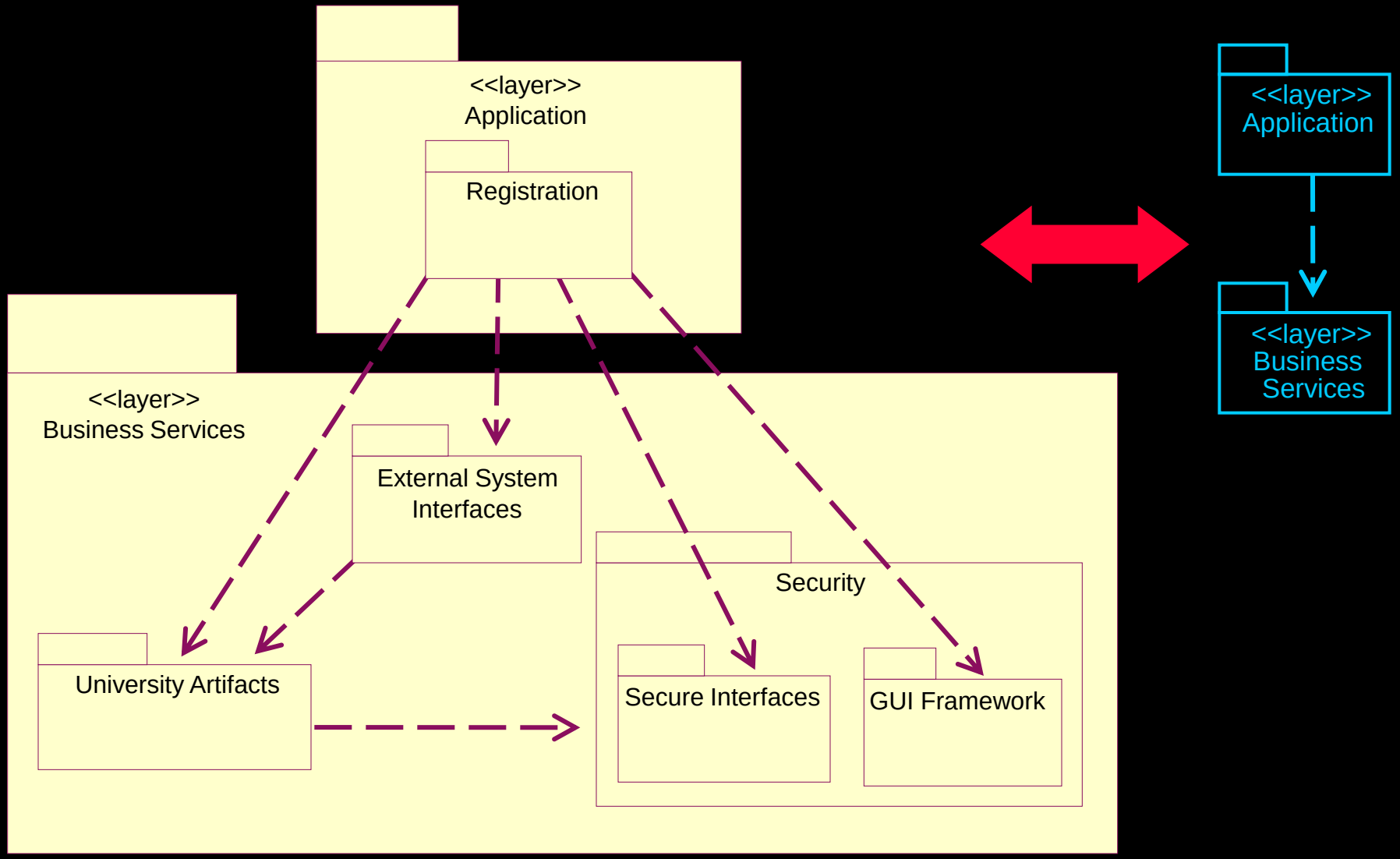
Example: Architectural Layers



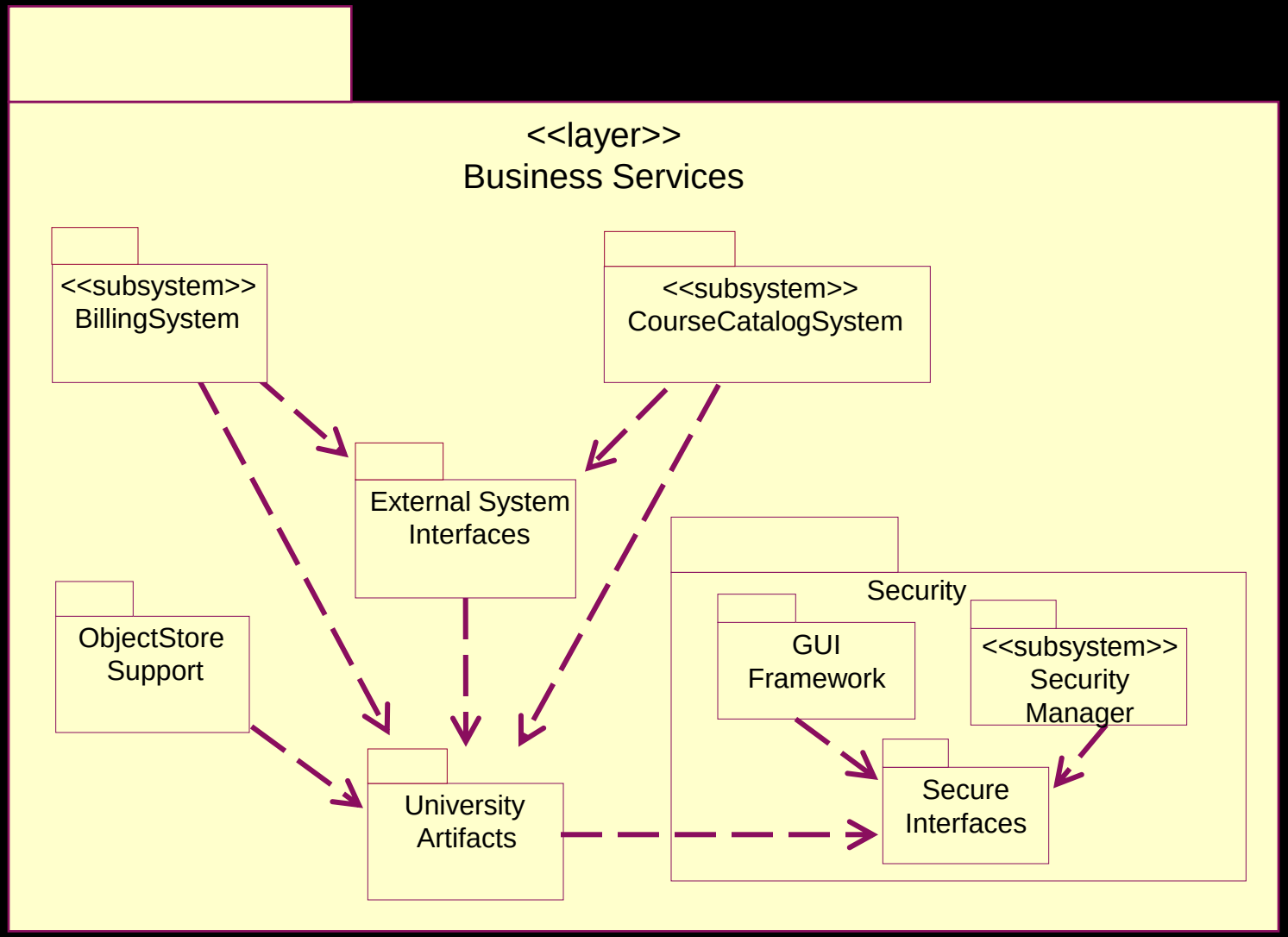
Example: Application Layer



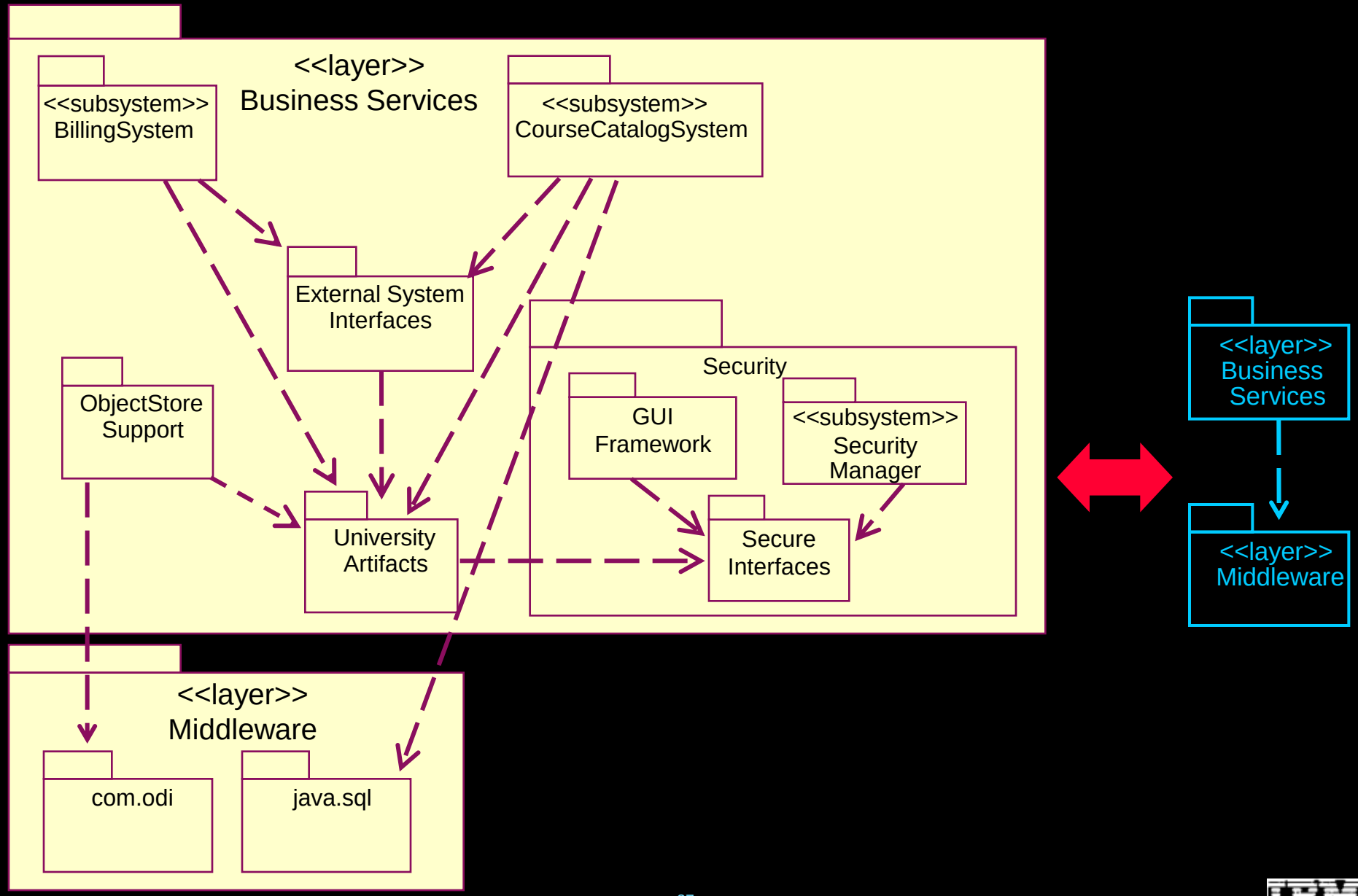
Example: Application Layer Context



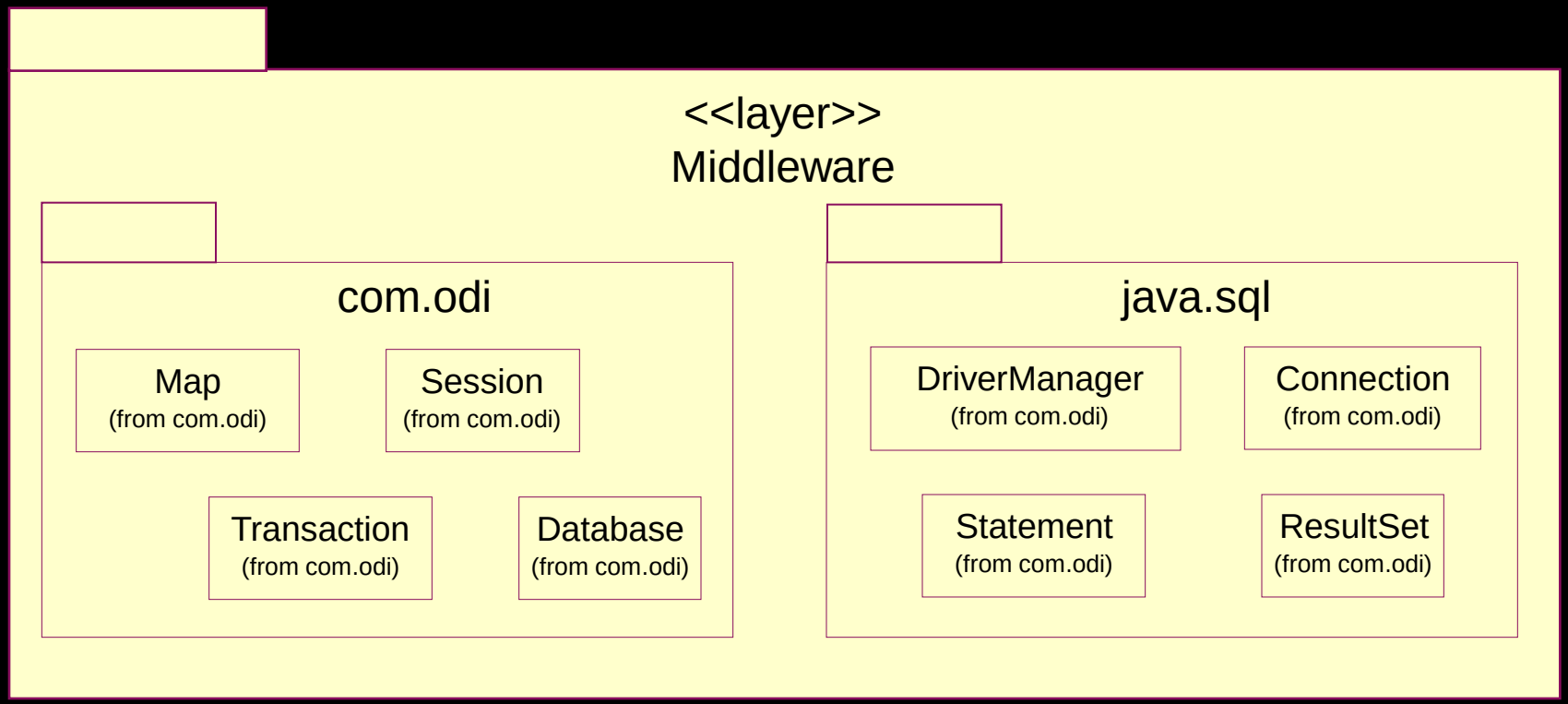
Example: Business Services Layer



Example: Business Services Layer Context



Example: Middleware Layer



7 RunTime Architecture

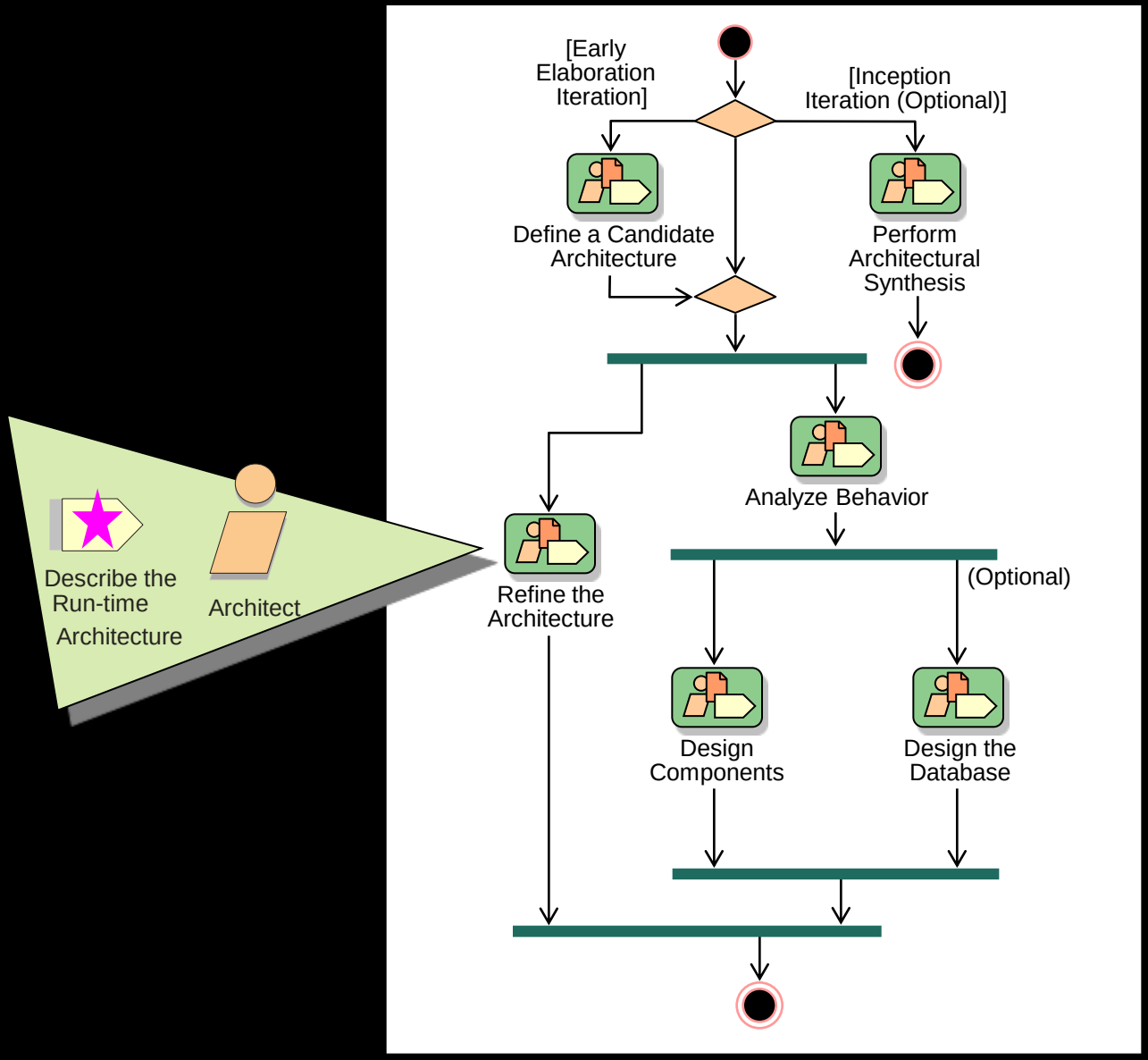
IBM Software Group

Mastering Object-Oriented Analysis and Design with UML

Module 7: Describe the Run-time Architecture

Rational. software

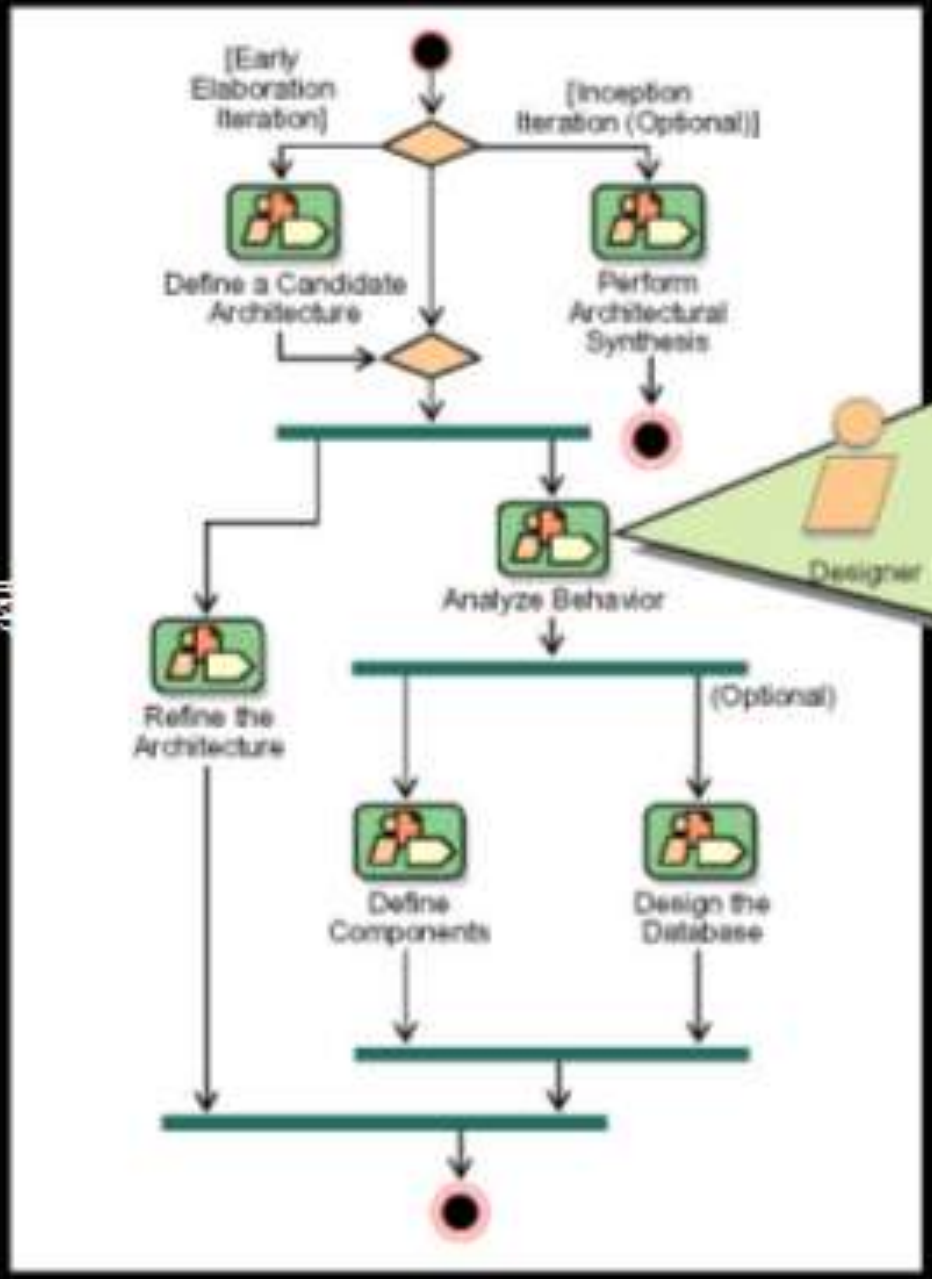
Describe the Run-time Architecture in Context



架构分析

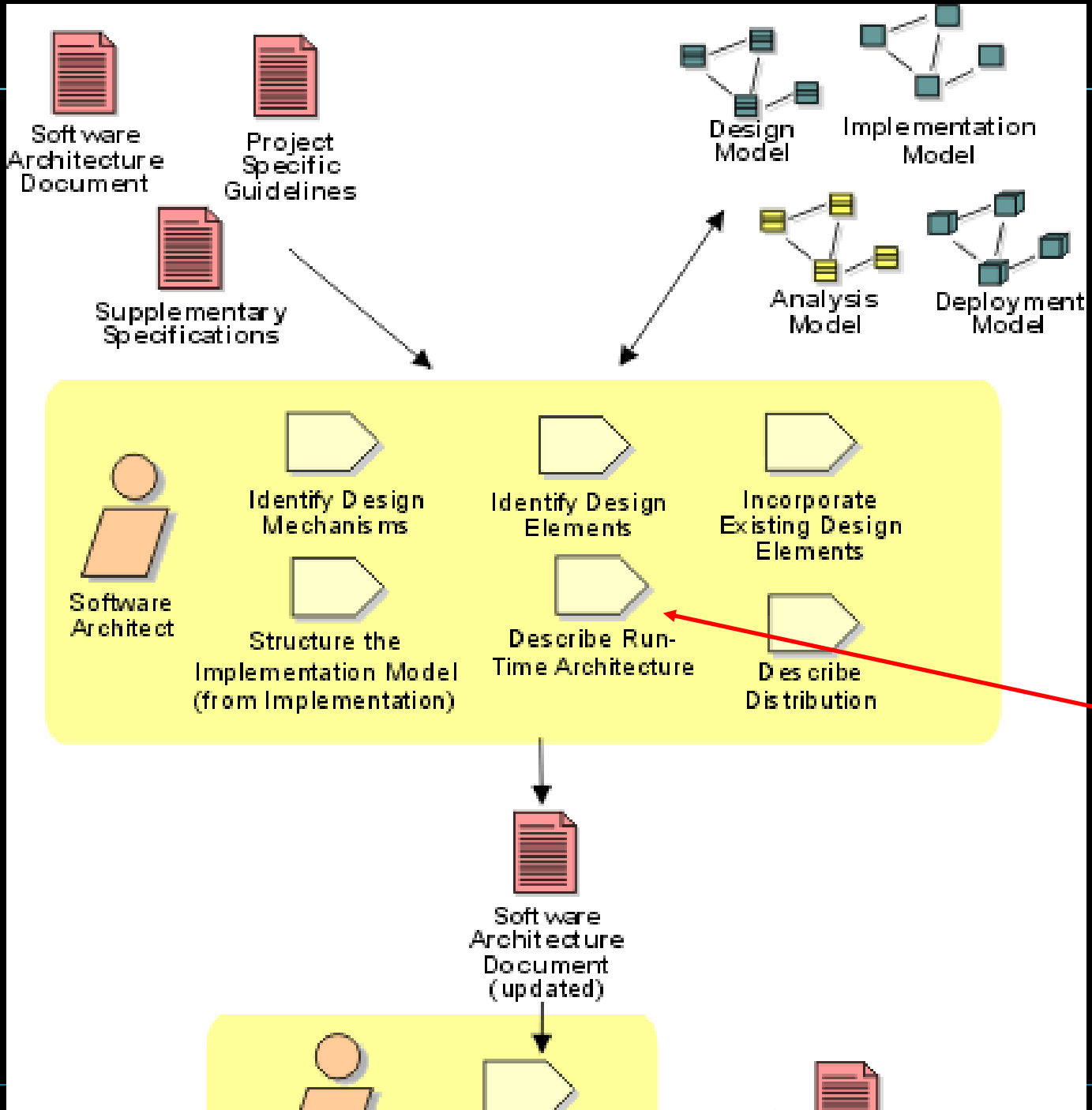
识别设计元素
运行时架构
描述分布

用例设计
子系统设计
类设计



用例分析

数据库设计


Software Architecture Document
Project Specific Guidelines
Supplementary Specifications
Design Model
Implementation Model
Analysis Model
Deployment Model
Software Architect
Identify Design Mechanisms
Identify Design Elements
Incorporate Existing Design Elements
Structure the Implementation Model (from Implementation)
Describe Run-Time Architecture
Describe Distribution
Software Architecture Document (updated)

Describe the Run-time Architecture Steps

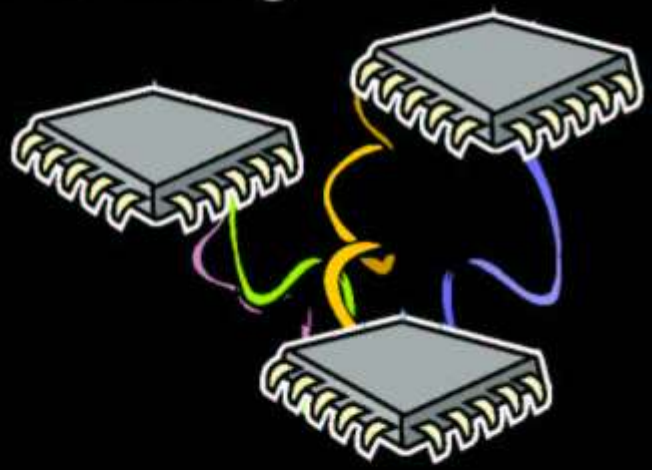
- ★ Analyze concurrency requirements
  - Identify processes and threads
  - Identify process lifecycles
  - Map processes onto the implementation
  - Distribute model elements among processes

Example: Concurrency Requirements

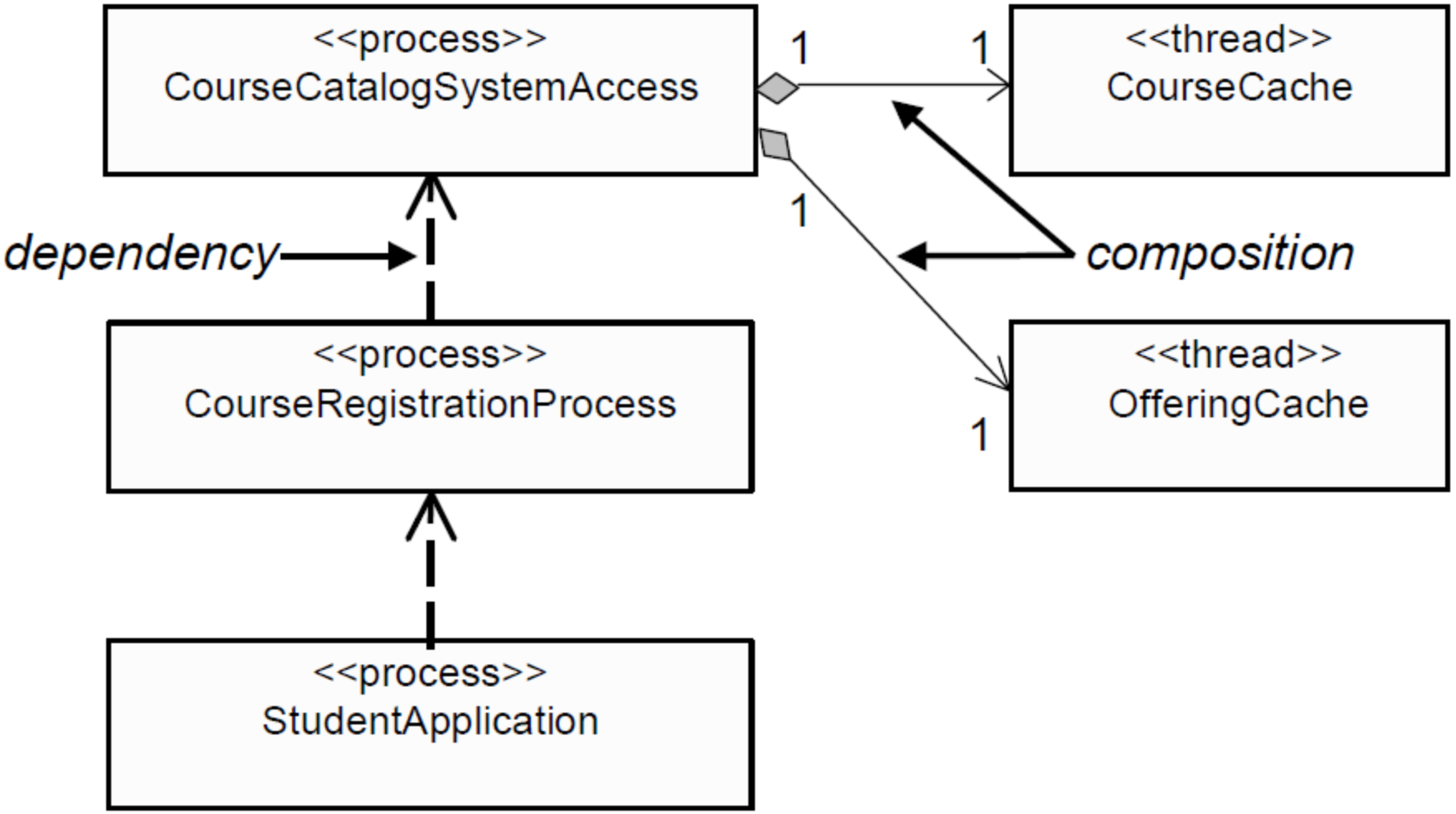
- In the Course Registration System, the concurrency requirements come from the requirements and the architecture:
  - Multiple users must be able to perform their work concurrently
  - If a course offering becomes full while a student is building a schedule including that offering, the student must be notified
  - Risk-based prototypes have found that the legacy course catalog database cannot meet our performance needs without some creative use of mid-tier processing power

Describe the Run-time Architecture Steps

- Analyze concurrency requirements
- ★ Identify processes and threads
- Identify process lifecycles
- Map processes onto the implementation
- Distribute model elements among processes



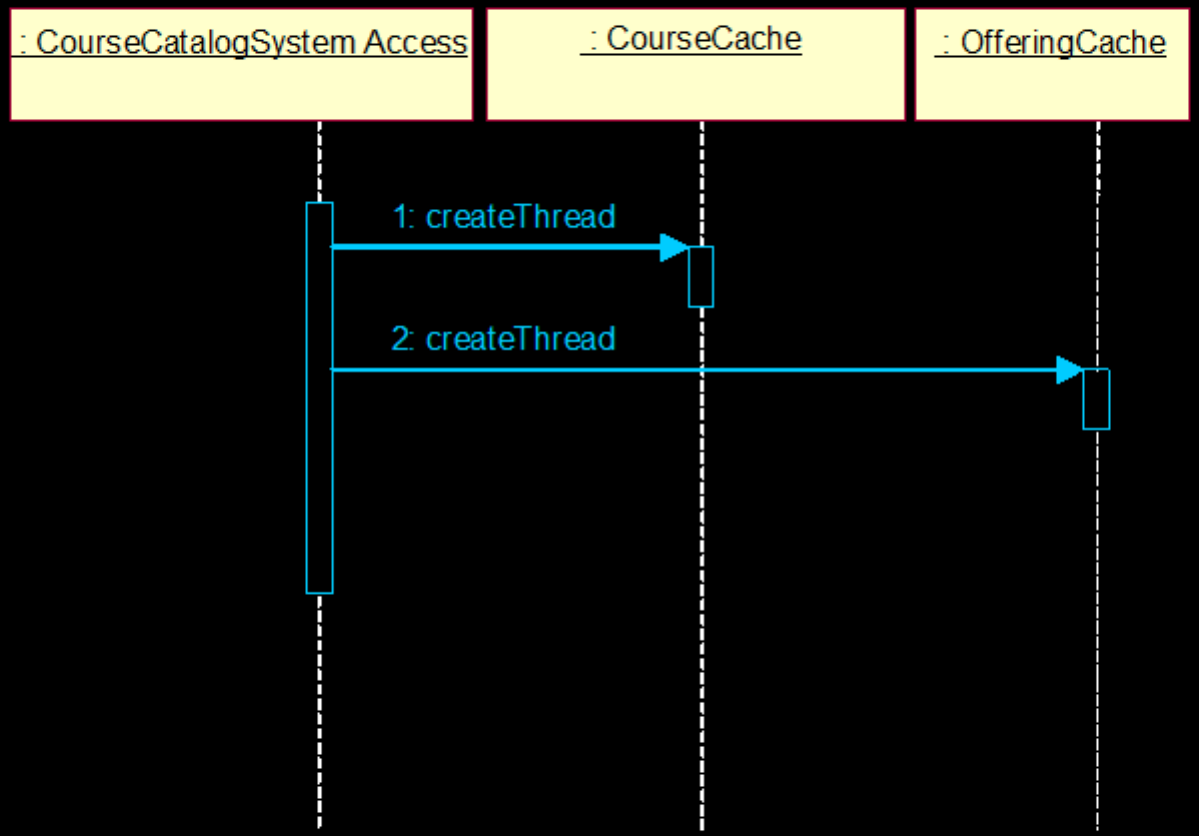
Example: Modeling Processes: Class Diagram



Describe the Run-time Architecture Steps

- Analyze concurrency requirements
- Identify processes and threads
- ★ Identify process lifecycles
- Map processes onto the implementation
- Distribute model elements among processes

Example: Create Processes and Threads



Creation of threads during application startup.

Describe the Run-time Architecture Steps

- Analyze concurrency requirements
- Identify processes and threads
- Identify process lifecycles
- ★ Map processes onto the implementation
- Distribute model elements among processes

Mapping Processes onto the Implementation

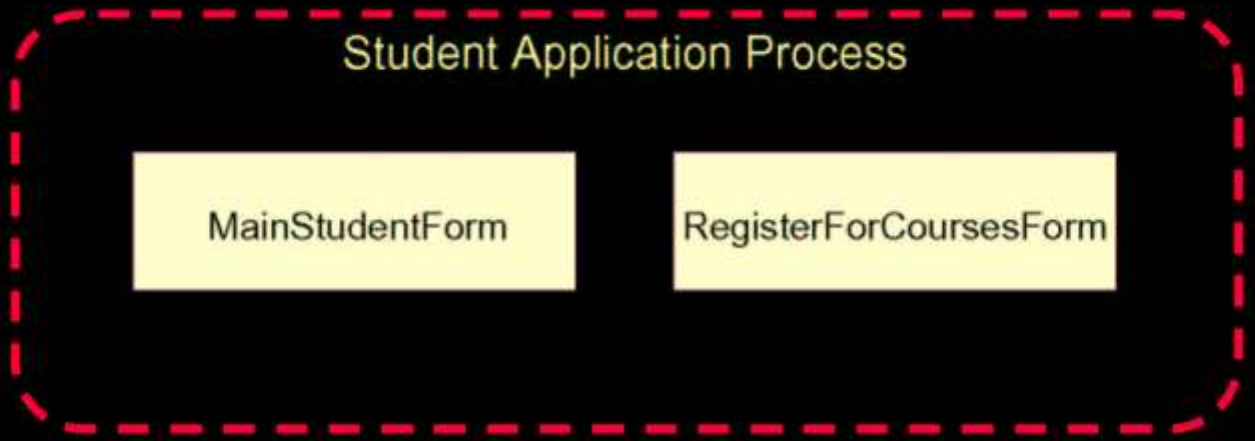
- Processes and threads must be mapped onto specific implementation constructs
- Considerations
  - Process coupling
  - Performance requirements
  - System process and thread limits
  - Existing threads and processes
  - IPC resource availability

Describe the Run-time Architecture Steps

- Analyze concurrency requirements
- Identify processes and threads
- Identify process lifecycles
- Map processes onto the implementation
- ★ Distribute model elements among processes

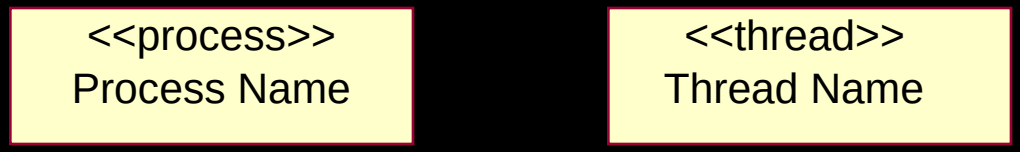
Design Element Allocation

- Instances of a given class or subsystem *must* execute within at least one process
  - They may execute in several processes

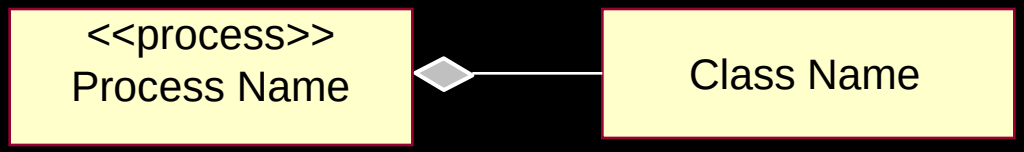


Modeling the Mapping of Elements to Processes

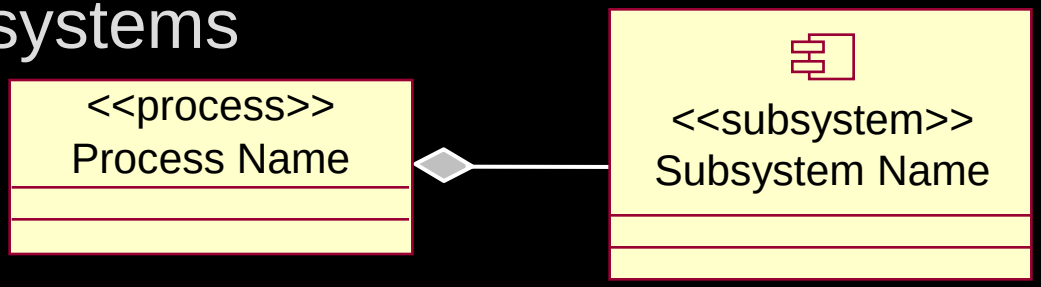
- Class diagrams
  - Active classes as processes/threads



  - Composition relationships from processes/threads to classes

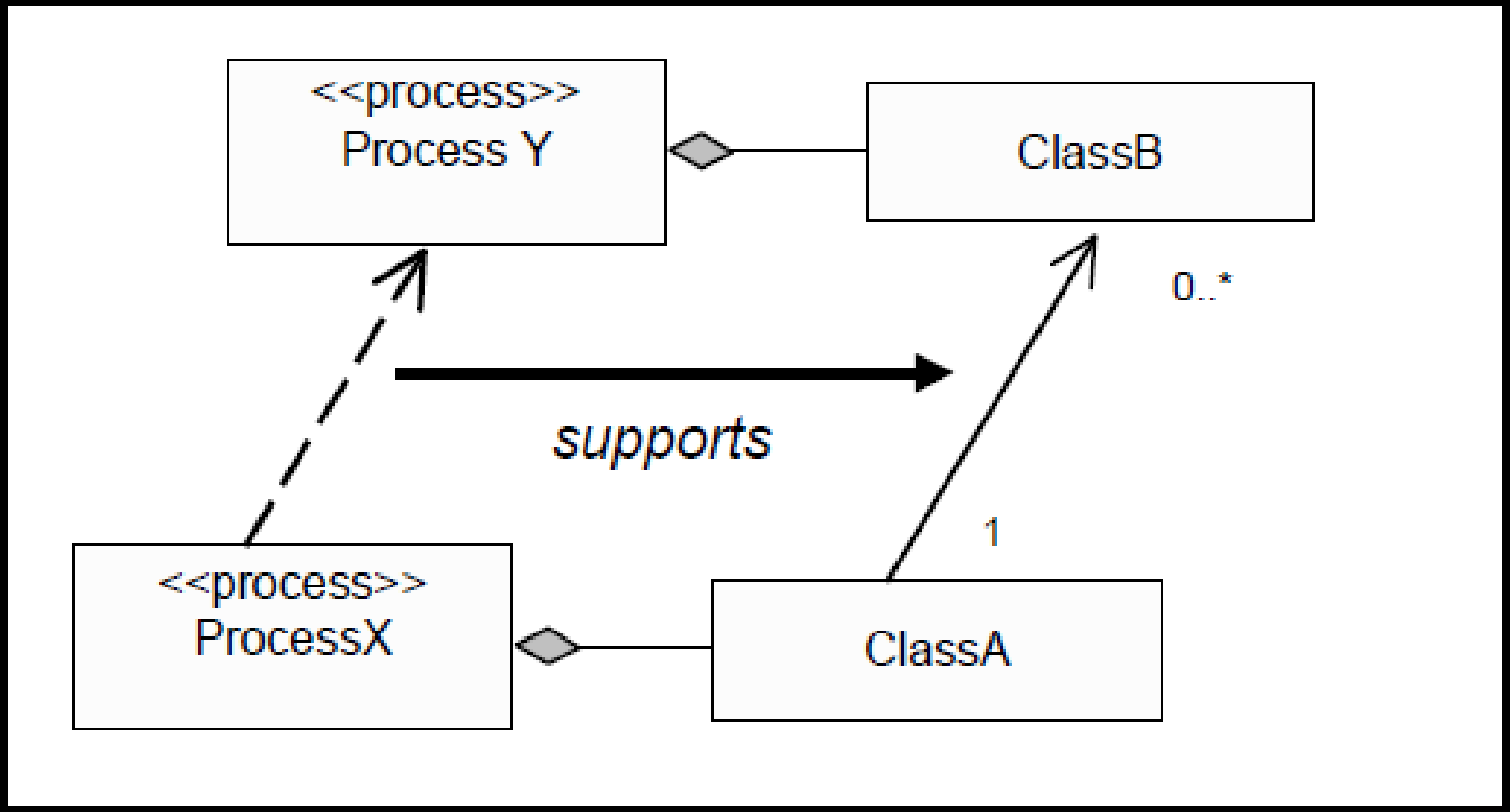


  - Composition relationships from processes/threads to subsystems

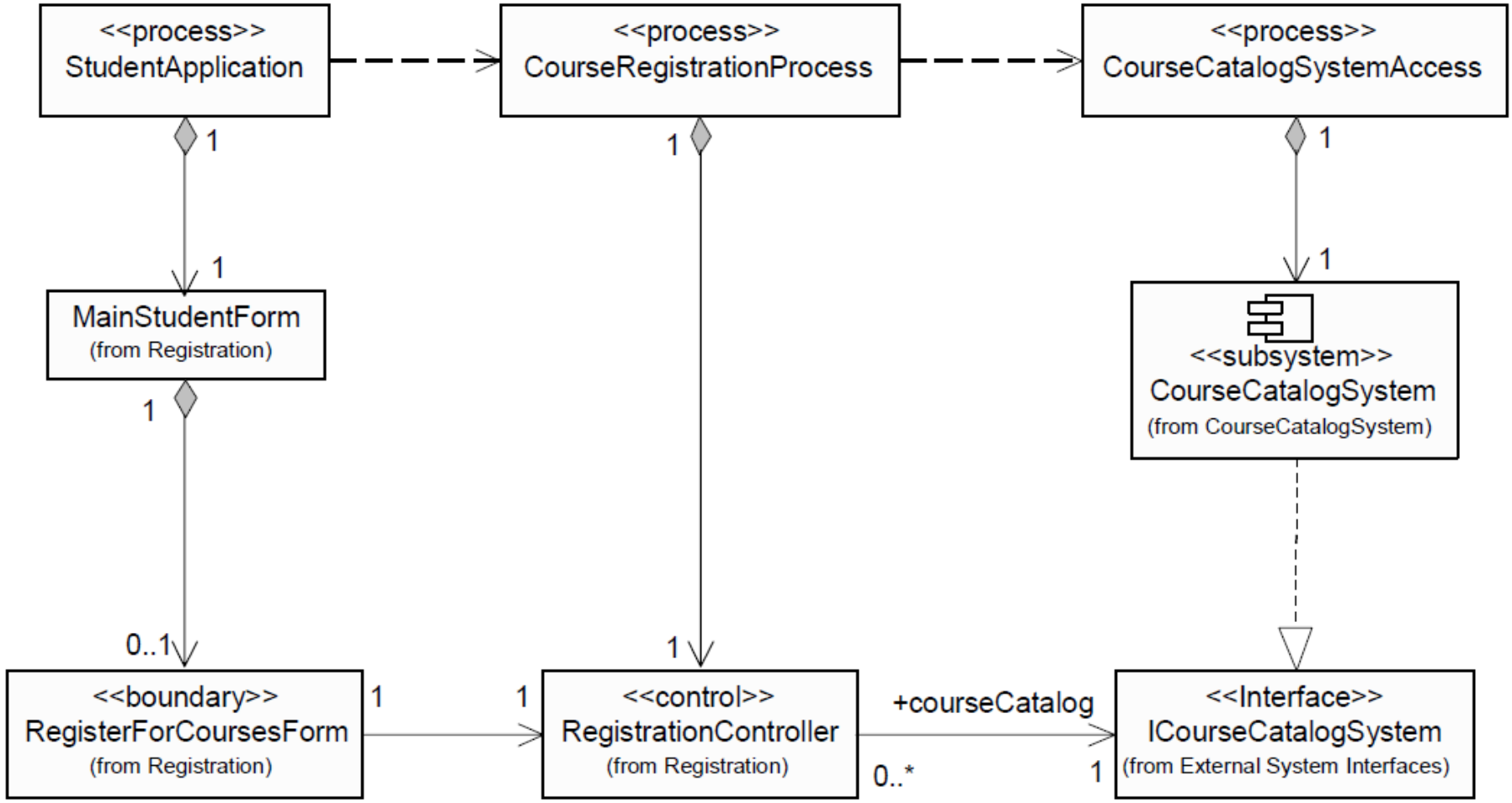


Process Relationships

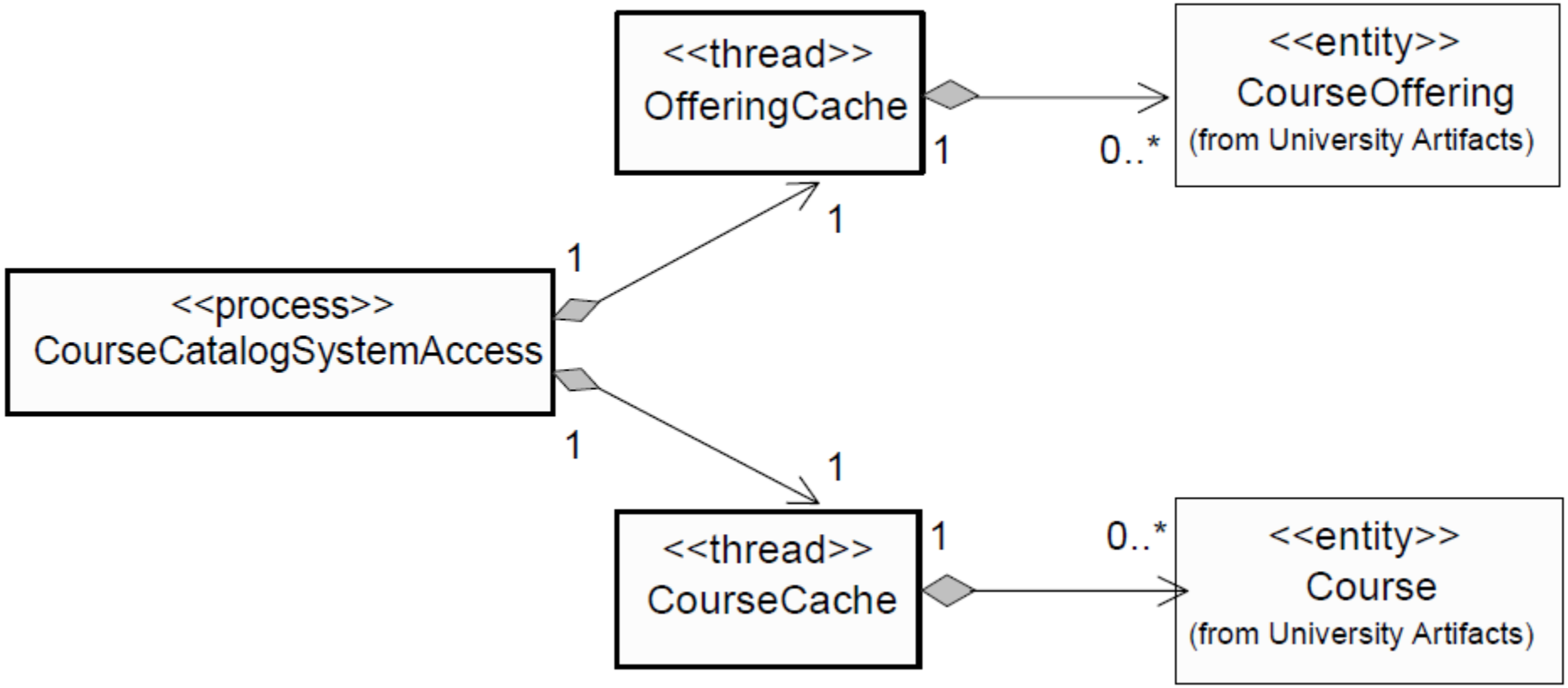
- Process relationships must support design element relationships



Example: Register for Course Processes



Example: Register for Course Processes (cont.)



8 Describe Distribution

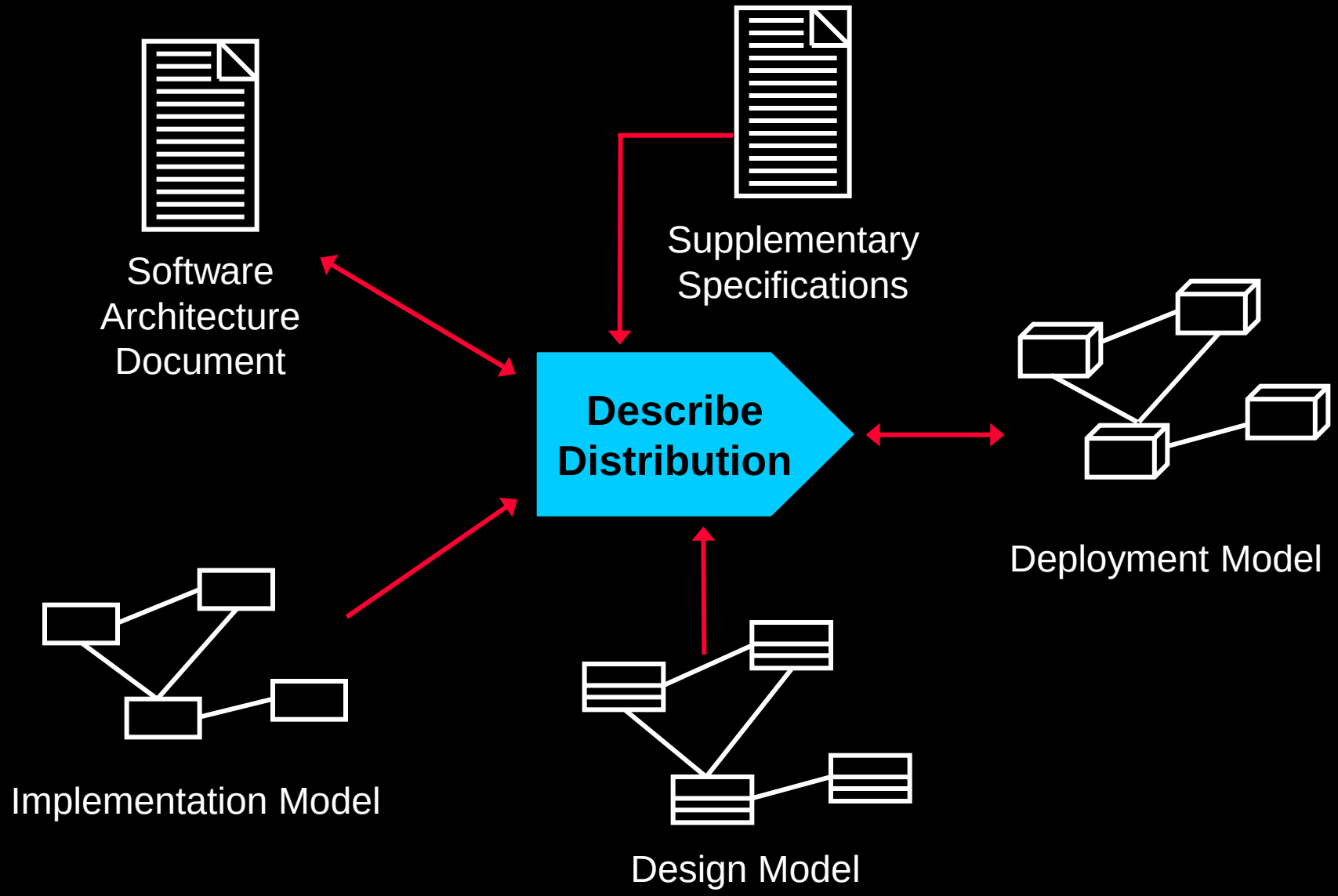
IBM Software Group

Mastering Object-Oriented Analysis and Design with UML

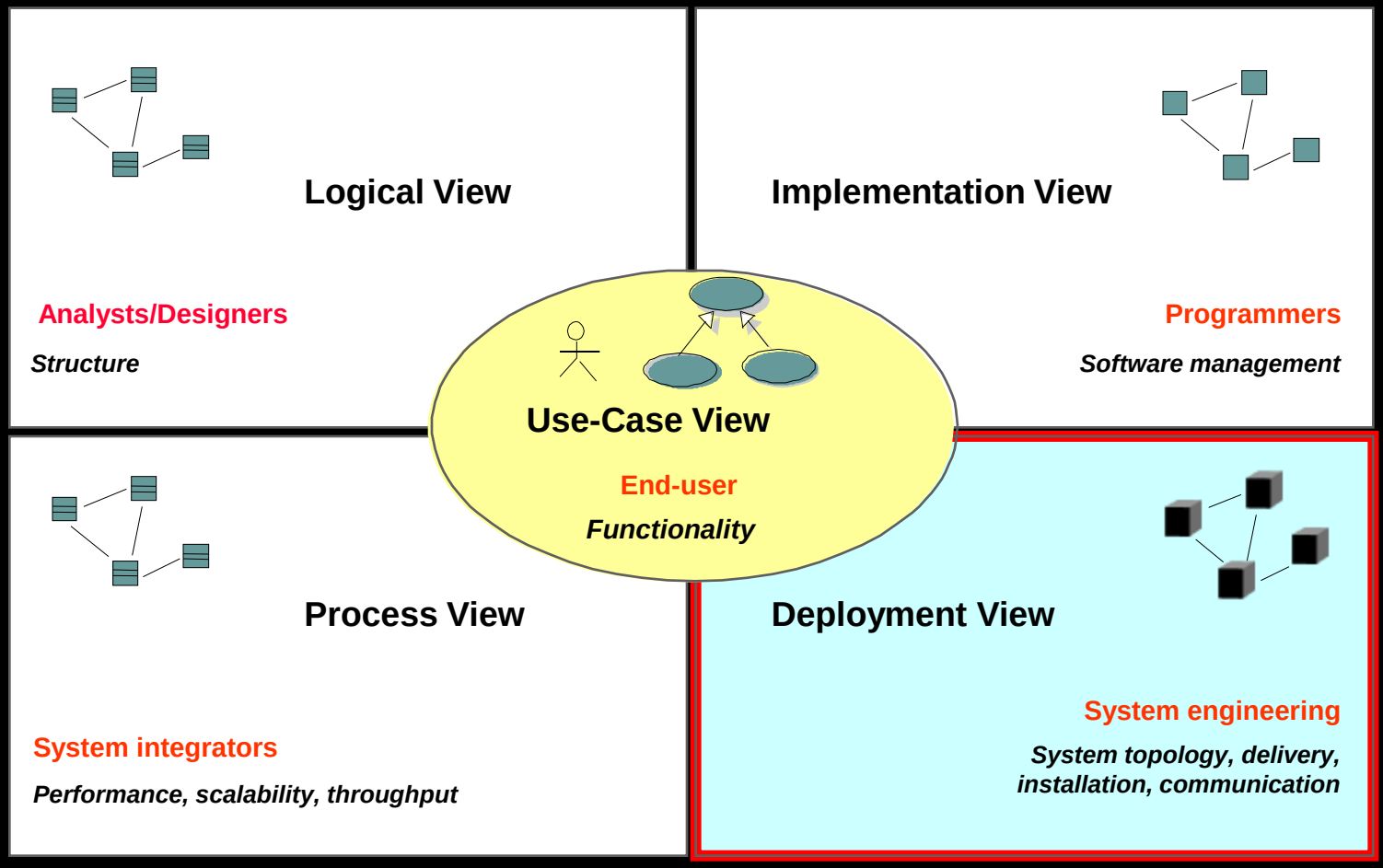
Module 8: Describe Distribution

Rational. software

Describe Distribution Overview

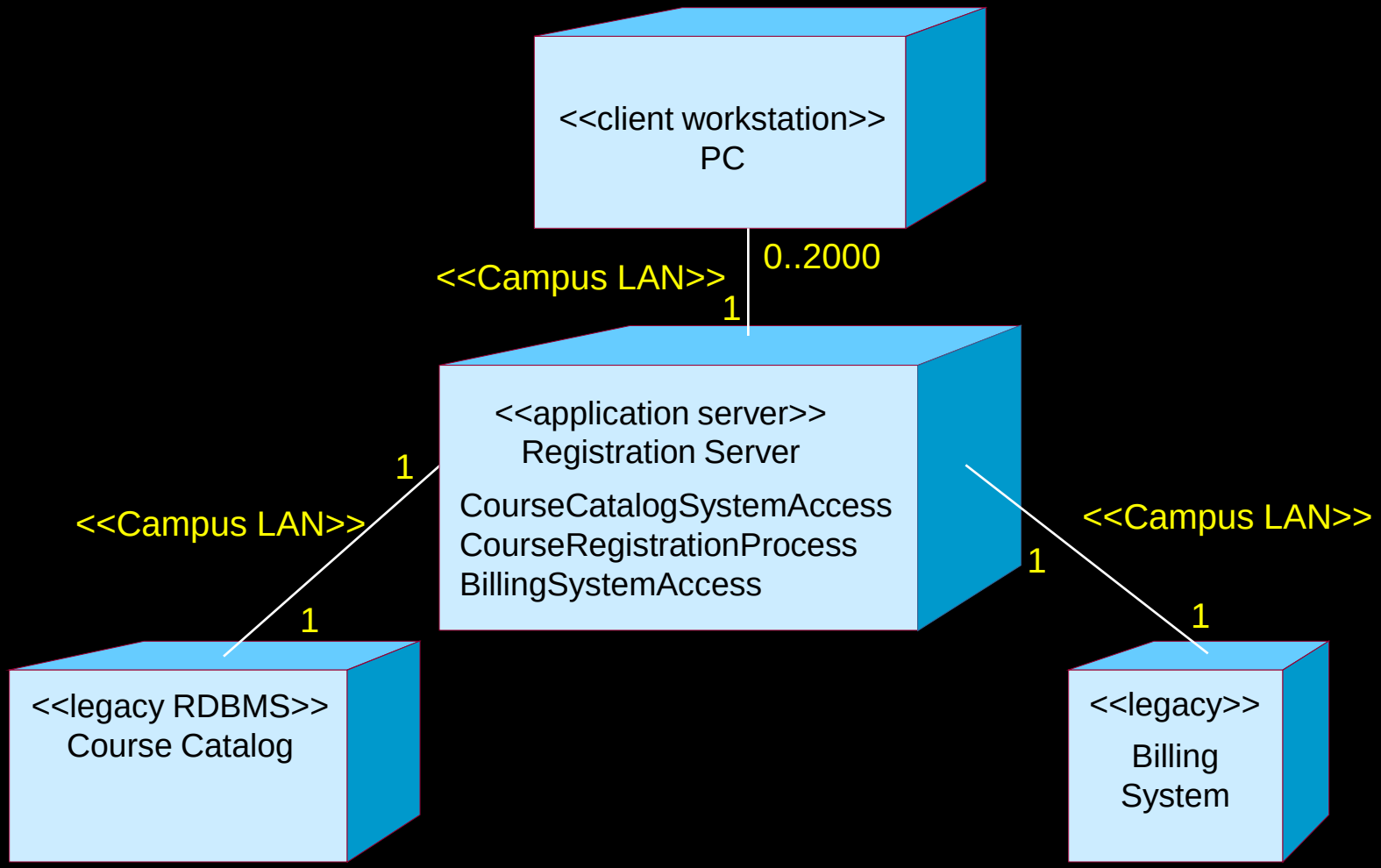


Key Concepts: The Deployment View



The Deployment View is an “architecturally significant” slice of the Deployment Model.

Review: Example: Deployment Diagram with Processes



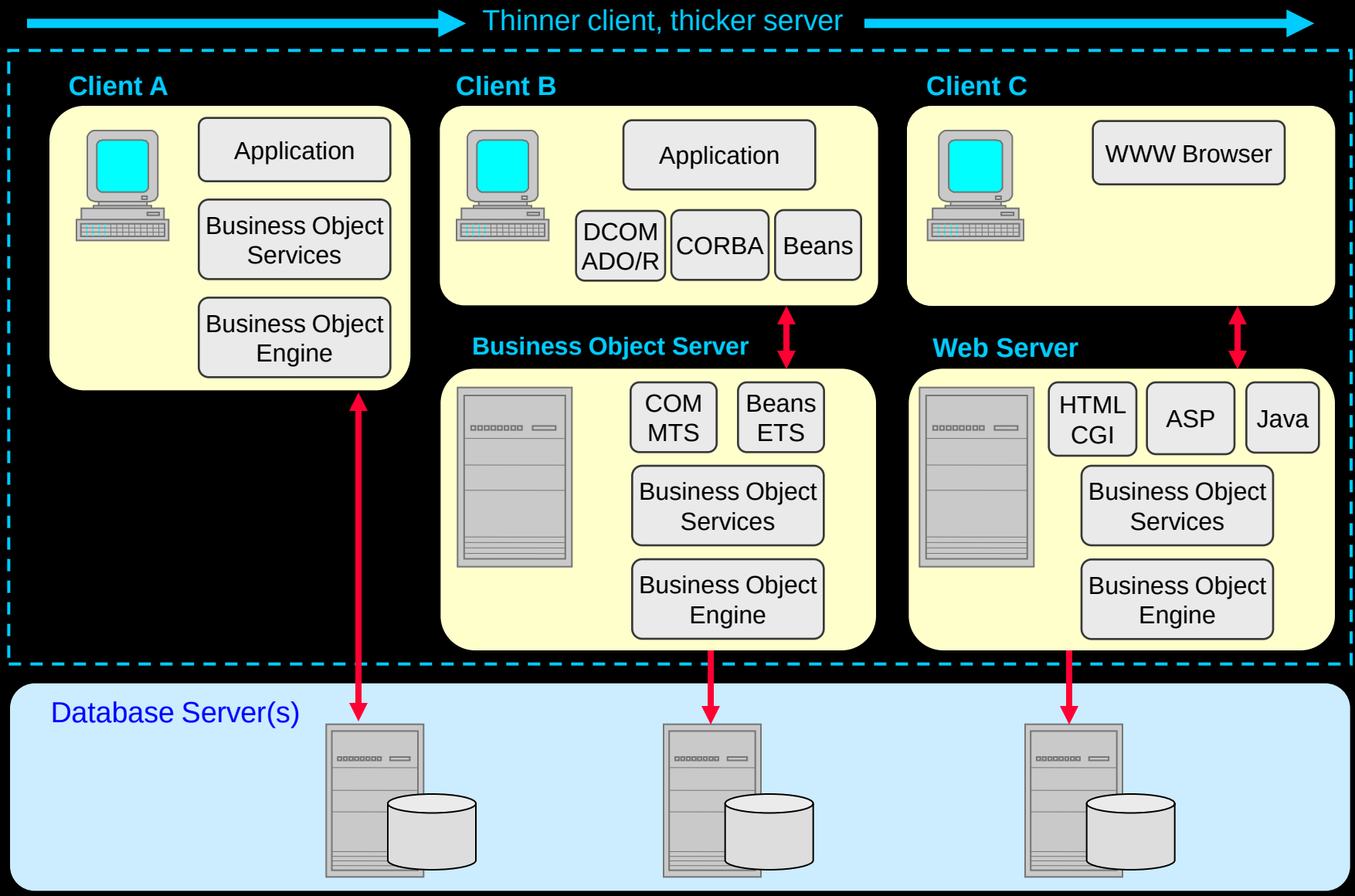
Why Distribute?

- Reduce processor load
- Special processing requirements
- Scaling concerns
- Economic concerns
- Distributed access to the system

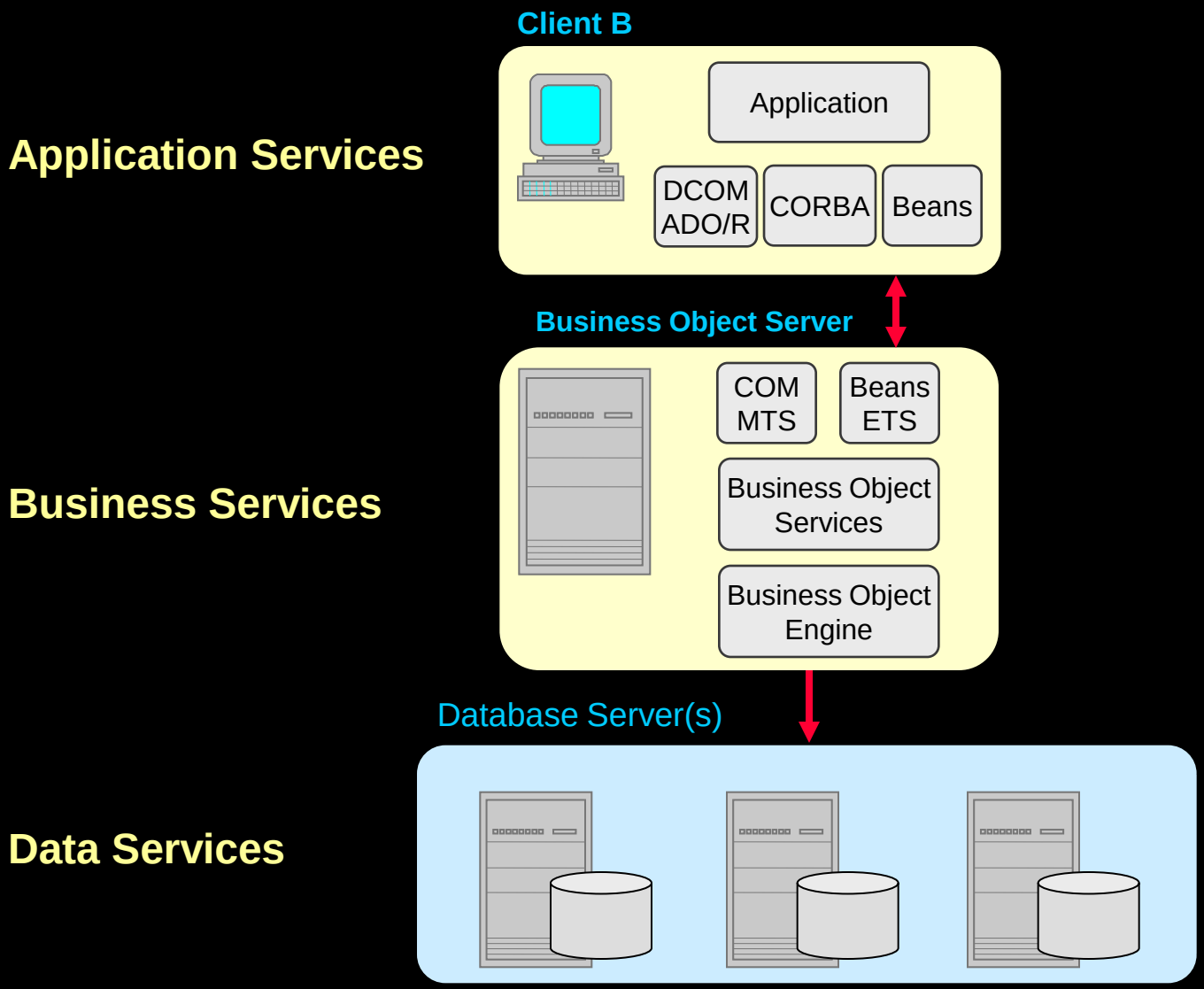
Distribution Patterns

- Client/Server
  - 3-tier
  - Fat Client
  - Fat Server
  - Distributed Client/Server
- Peer-to-peer

Client/Server Architectures



Client/Server: Three-Tier Architecture

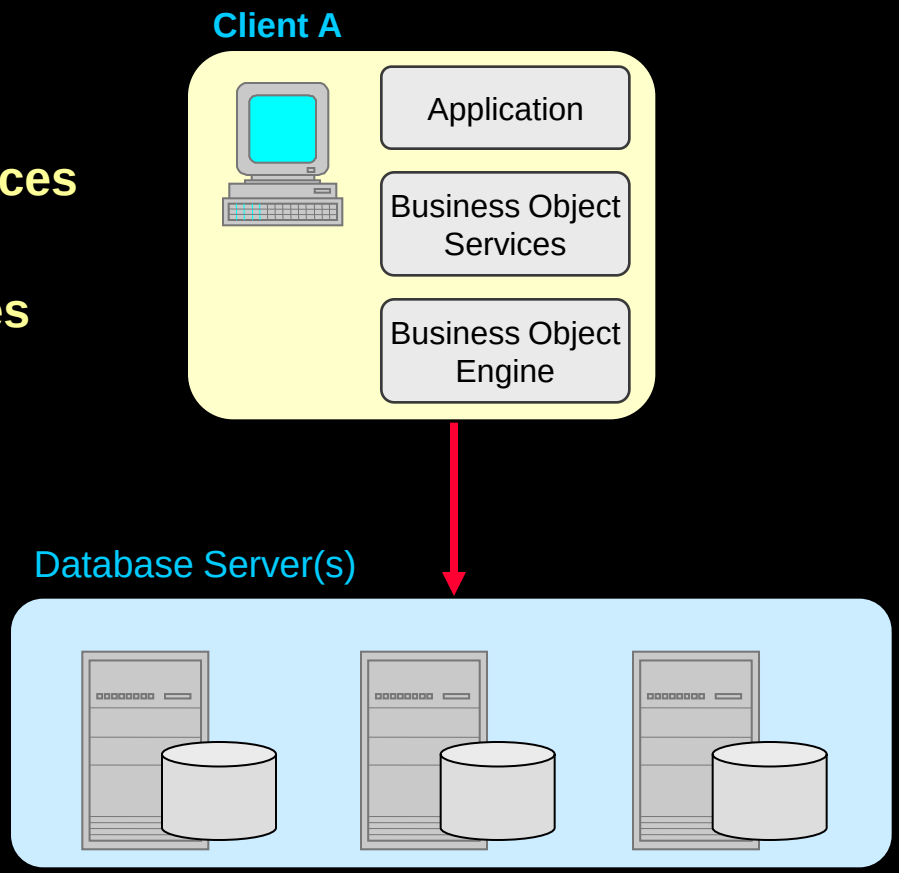


Client/Server: “Fat Client” Architecture

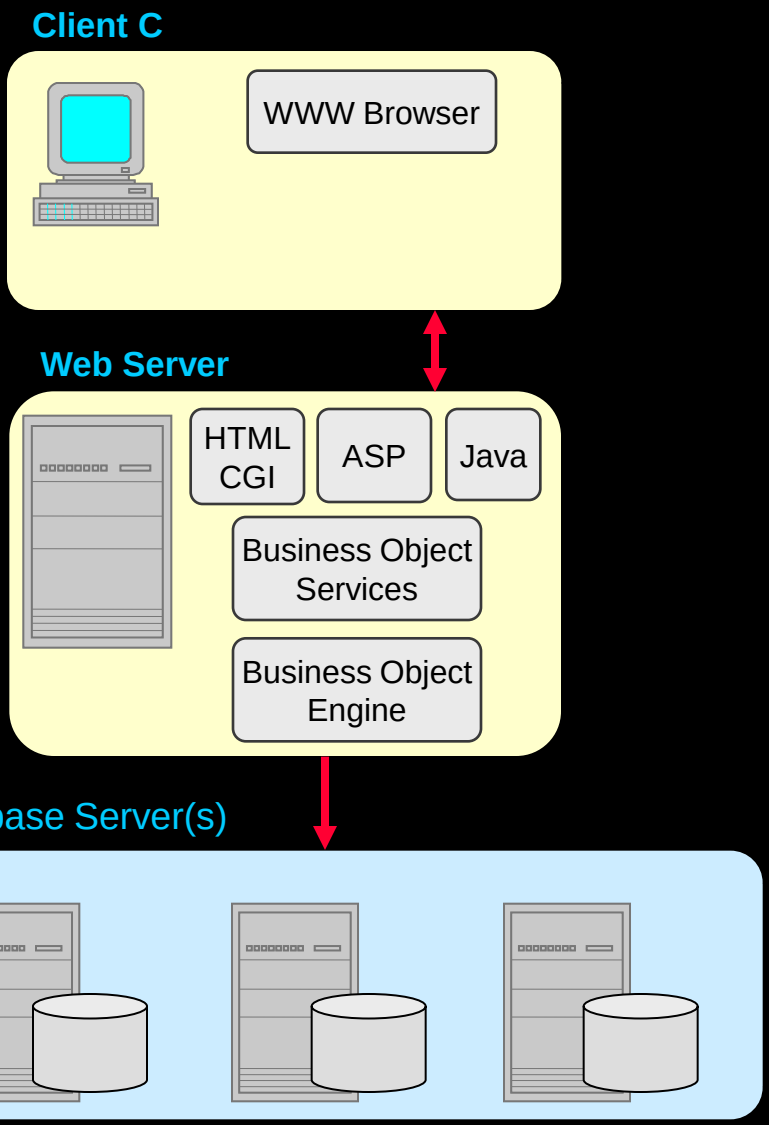
Application Services

Business Services

Data Services



Client/Server: Web Application Architecture

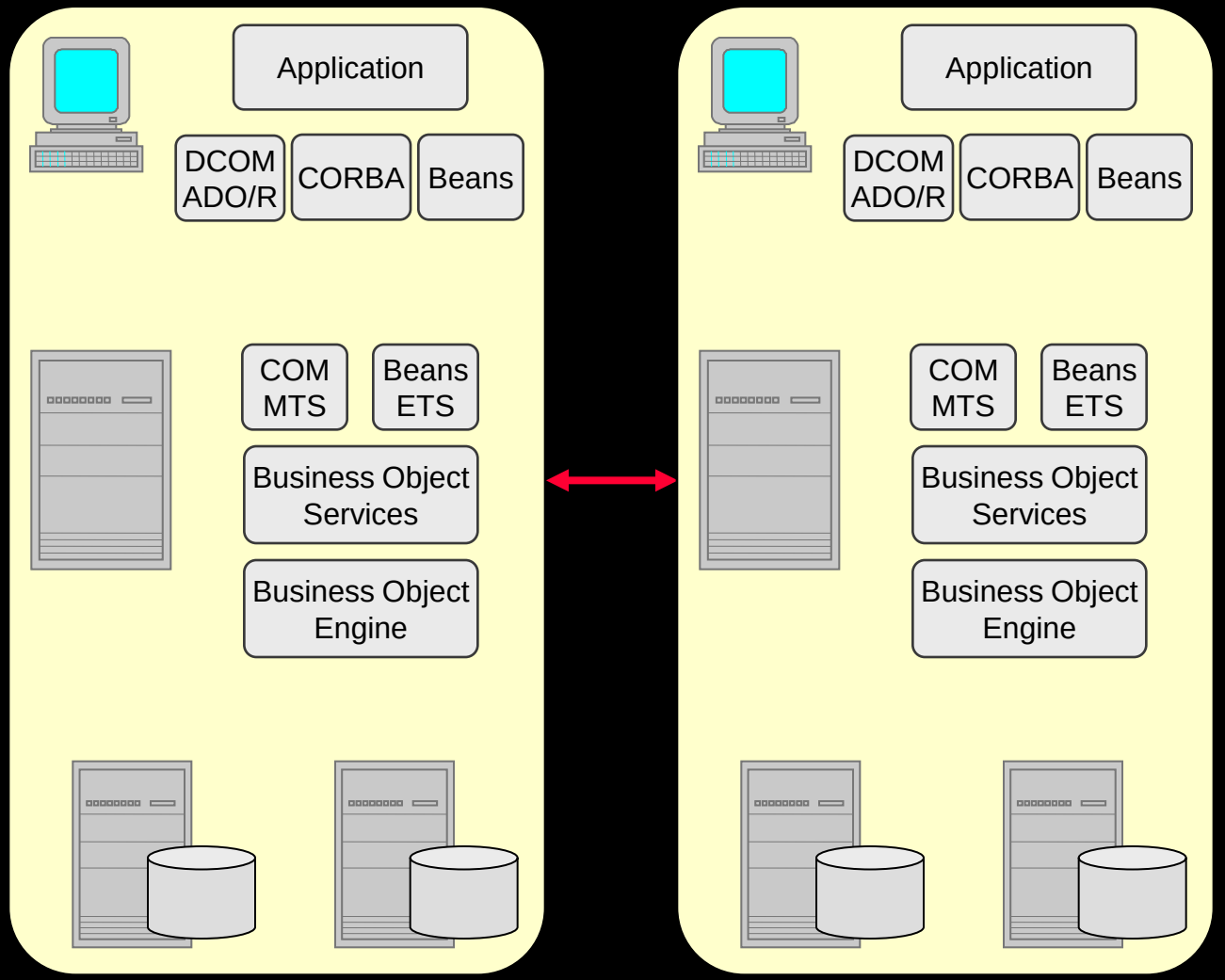


Application Services

Business Services

Data Services

Peer-to-Peer Architecture

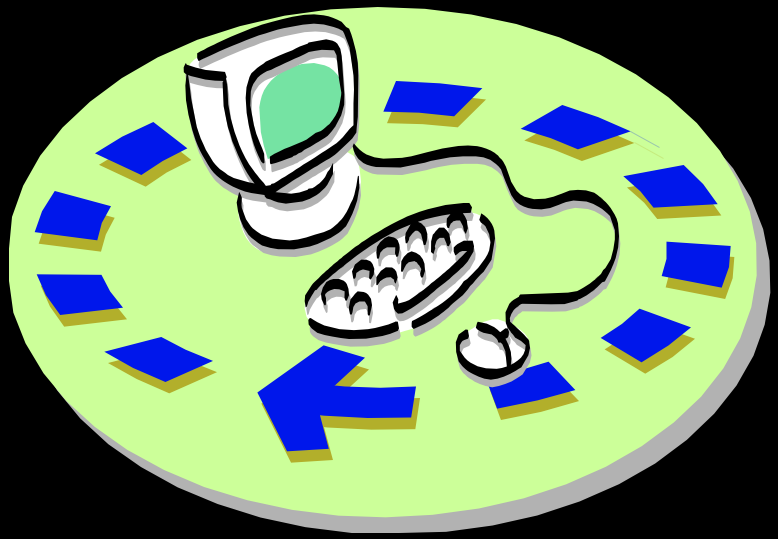


Describe Distribution Steps

- Define the network configuration
- Allocate processes to nodes
- Define the distribution mechanism

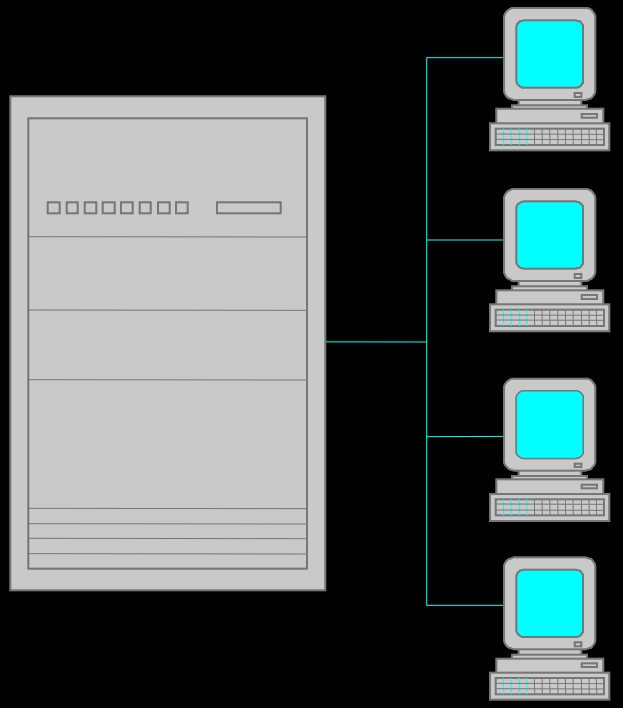
Describe Distribution Steps

- ★ Define the network configuration
- Allocate processes to nodes
- Define the distribution mechanism



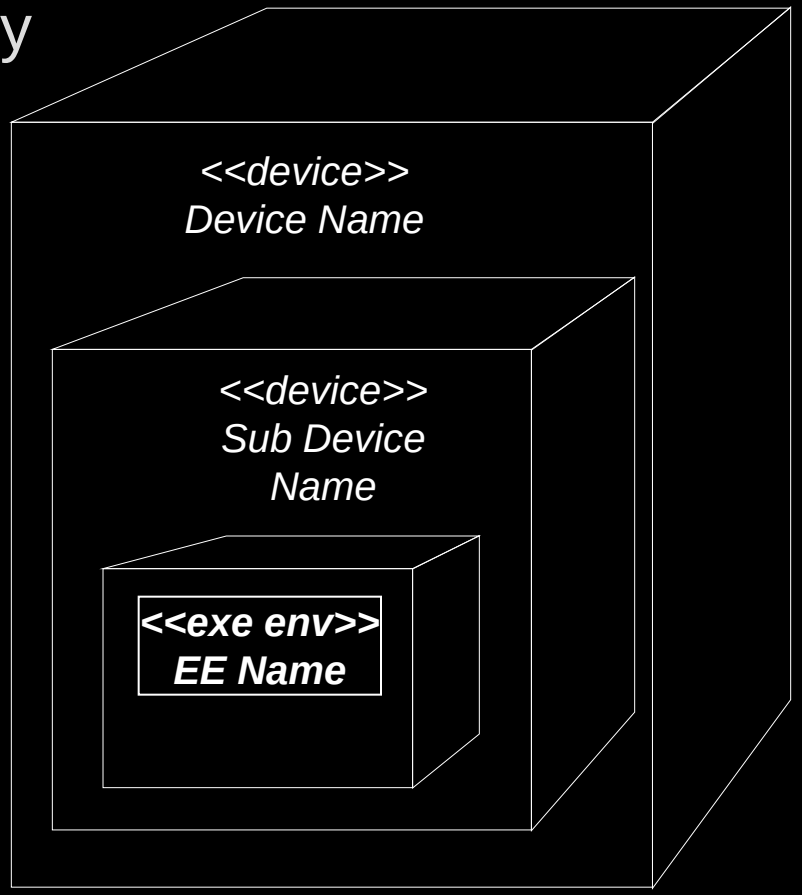
The Network Configuration

- End-user workstation nodes
- ”Headless" processing server nodes
- Special configurations
  - Development
  - Test
- Specialized processors



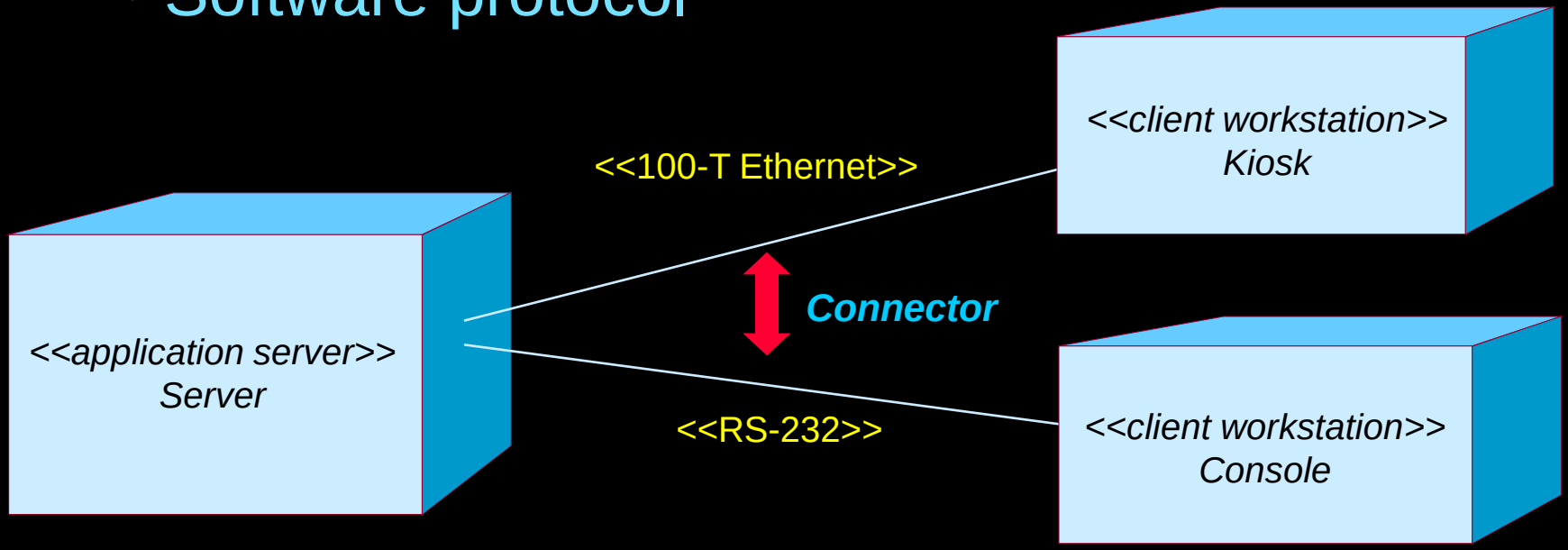
Review: What Is a Node?

- Represents a run-time computational resource
  - Generally has at least memory and often processing capability.
- Types:
  - Device
    - Physical computational resource with processing capability.
    - May be nested
  - Execution Environment
    - Represent particular execution platforms

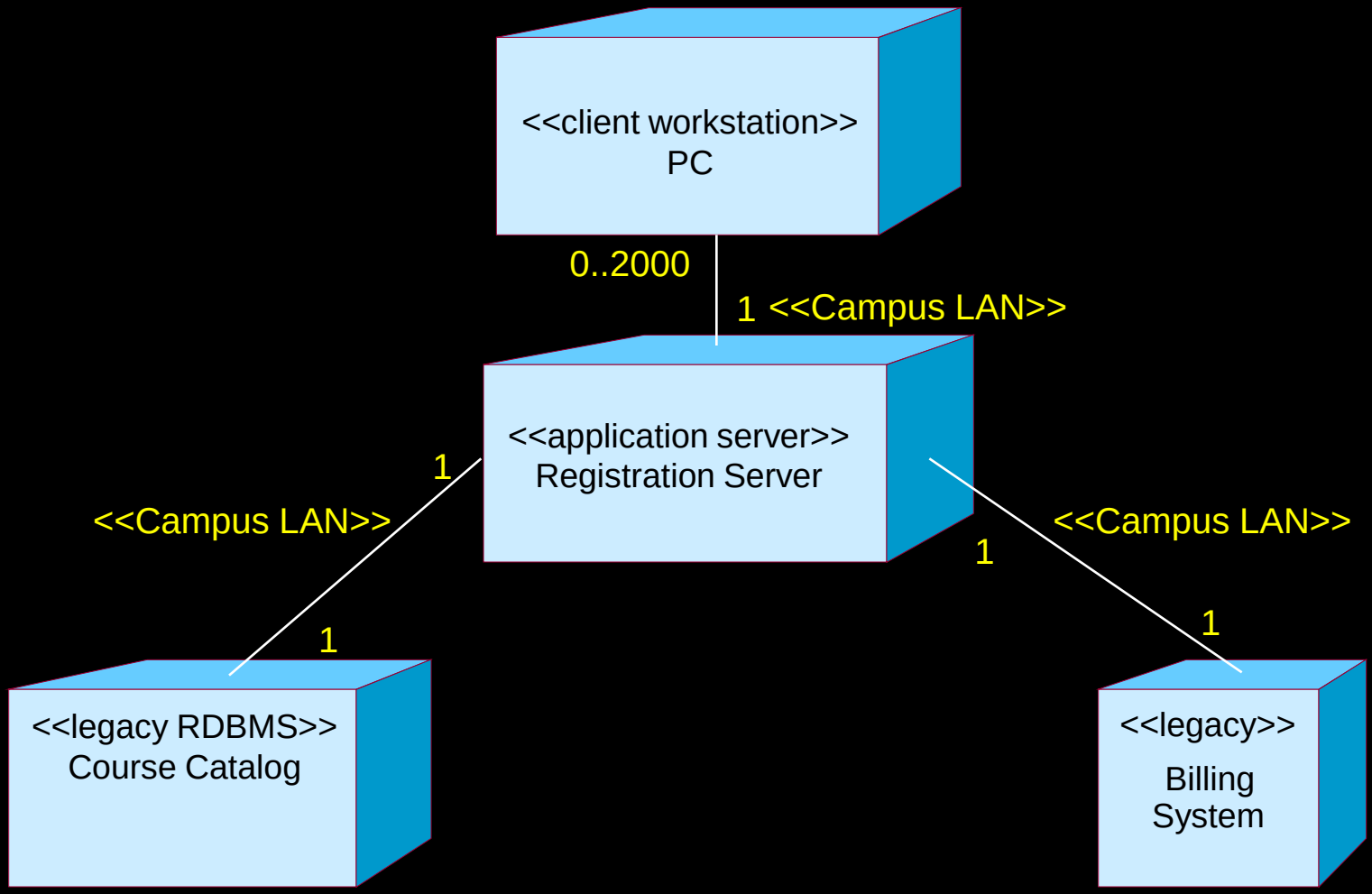


Review: What Is a Connector?

- A connector represents a:
  - Communication mechanism
    - Physical medium
    - Software protocol



Review: Example: Deployment Diagram

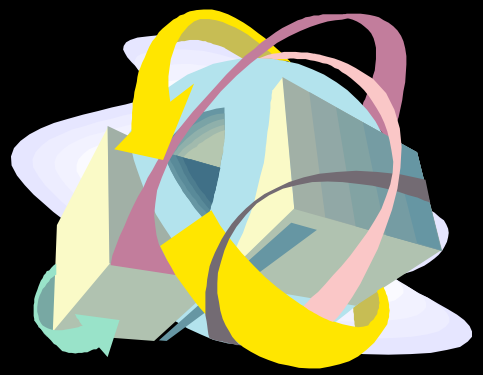


Describe Distribution Steps

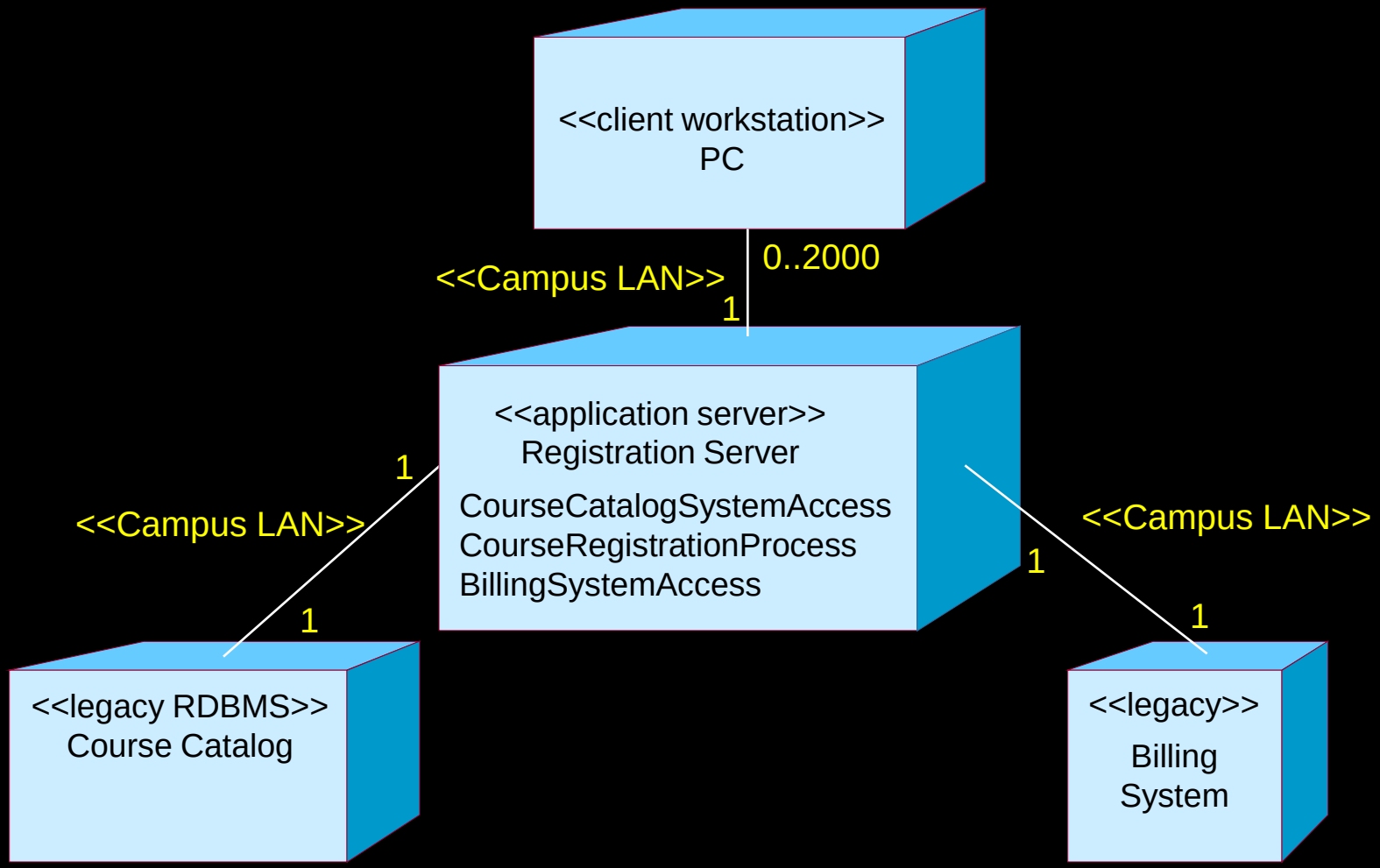
- Define the network configuration
- ★ Allocate processes to nodes
- Define the distribution mechanism

Process-to-Node Allocation Considerations

- Distribution patterns
- Response time and system throughput
- Minimization of cross-network traffic
- Node capacity
- Communication medium bandwidth
- Availability of hardware and communication links
- Rerouting requirements



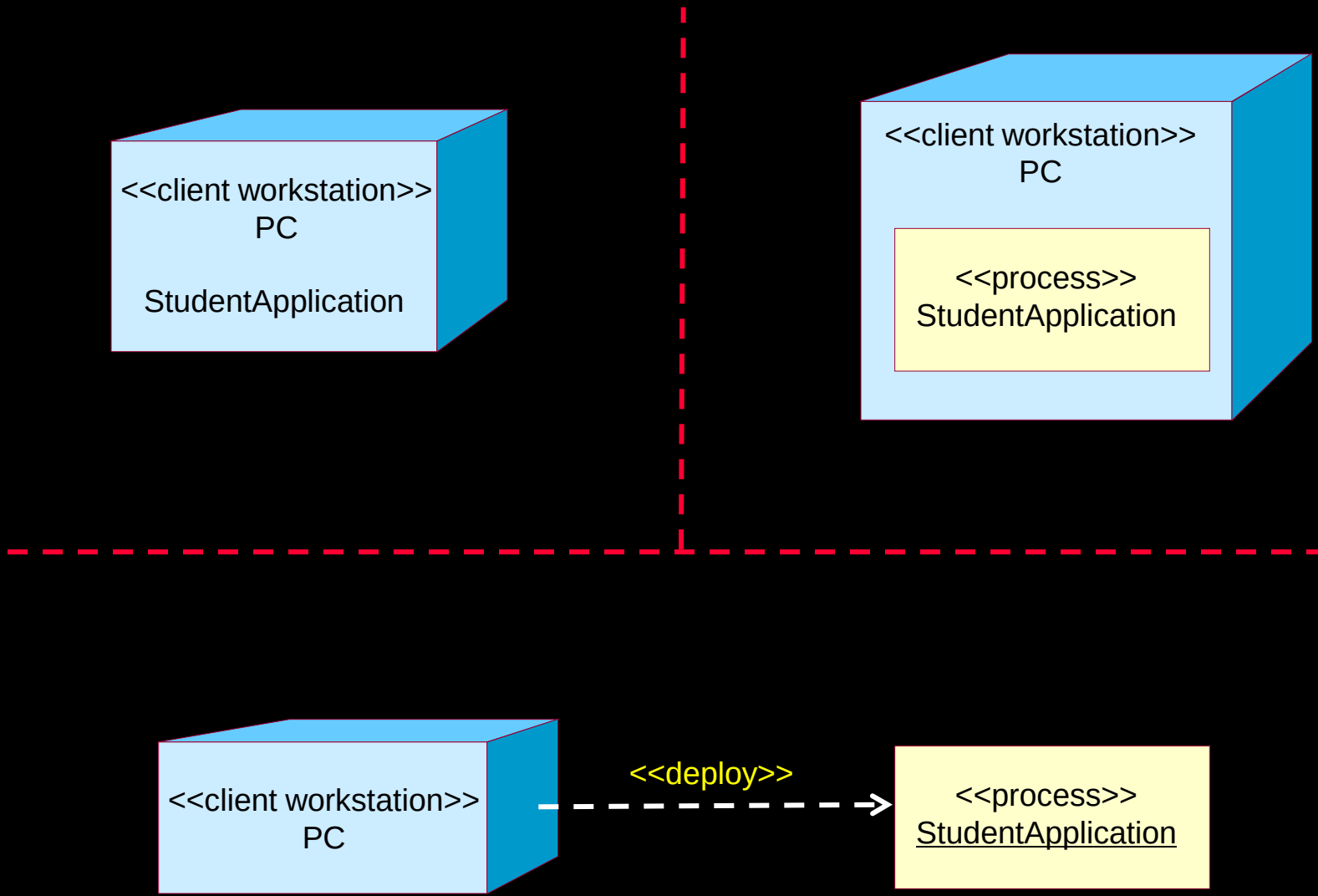
Review: Example: Deployment Diagram with Processes



What is Deployment?

- Deployment is the assignment, or mapping, of software artifacts to physical nodes during execution.
  - Artifacts are the entities that are deployed onto physical nodes
    - Processes are assigned to computers
- Artifacts model physical entities.
  - Files, executables, database tables, web pages, etc.
- Nodes model computational resources.
  - Computers, storage units.

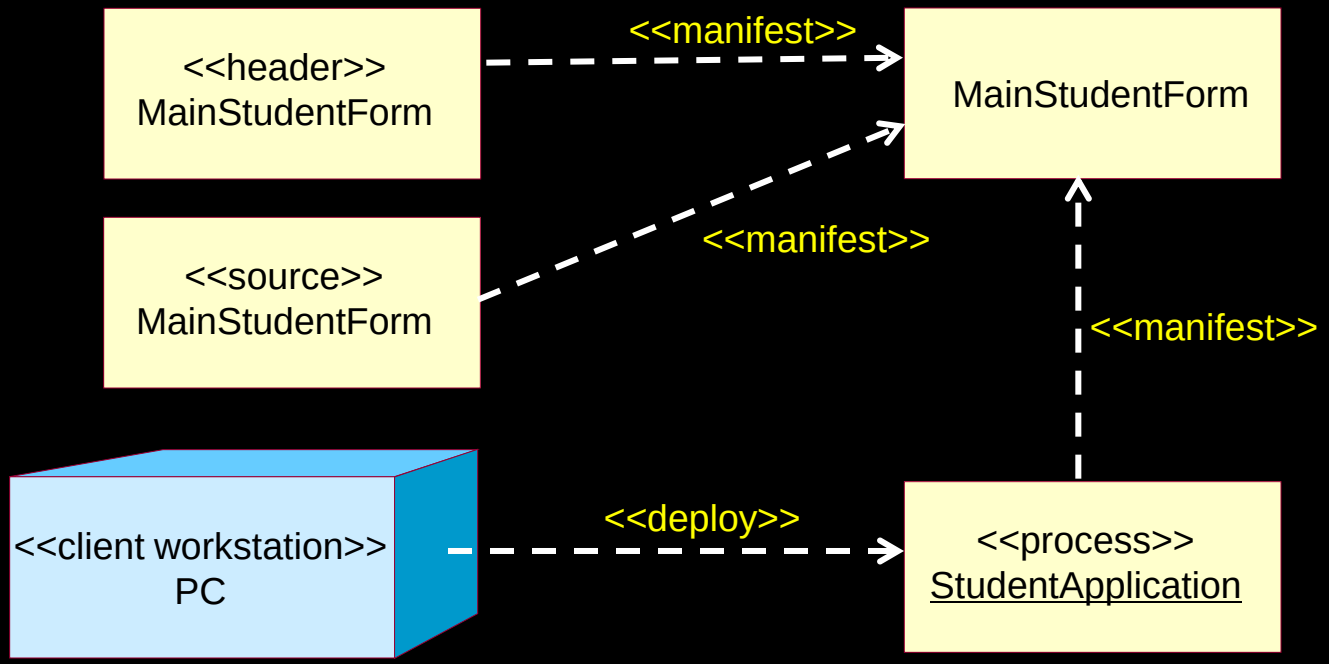
Example: Deploying Artifacts to Nodes



What is Manifestation?

- The physical implementation of a model element as an artifact.
  - A relationship between the model element and the artifact that implements it
    - Model elements are typically implemented as a set of artifacts.
      - Source files, executable files, documentation file

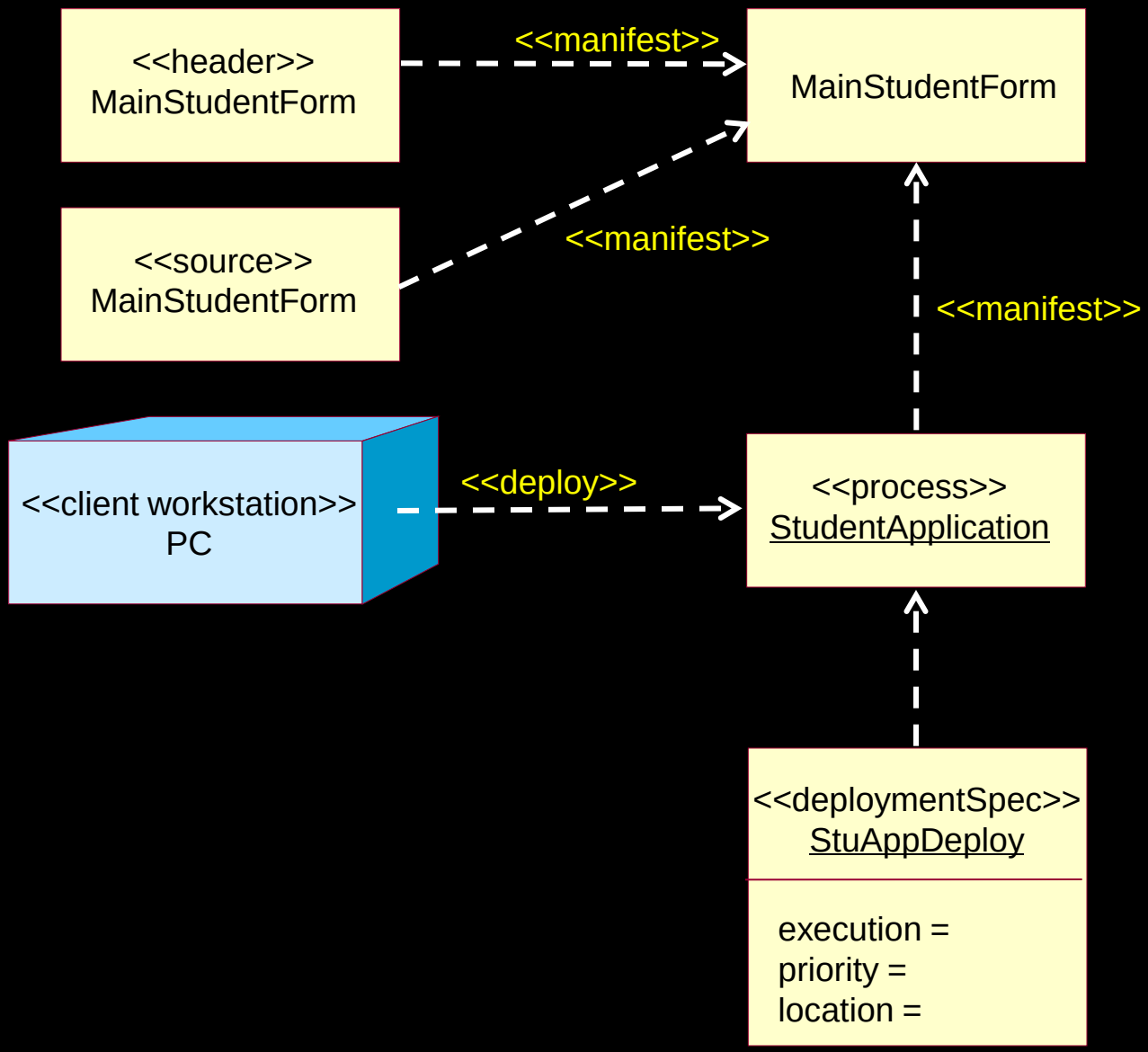
Example: Manifestation



What is a Deployment Specification?

- A detailed specification of the parameters of the deployment of an artifact to a node.
  - May define values that parameterize the execution

Example: Deployment Specification

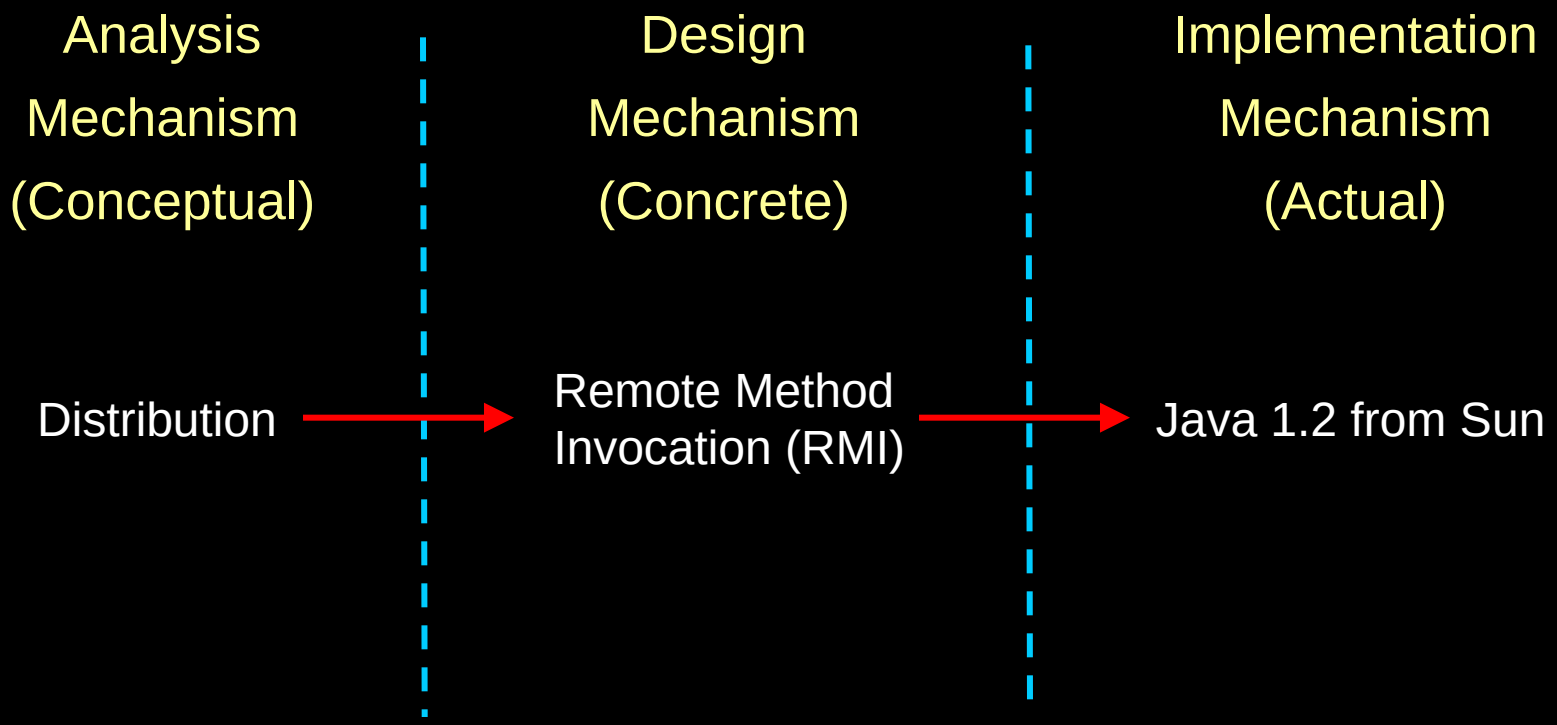


Describe Distribution Steps

- Define the network configuration
- Allocate processes to nodes
- ★ Define the distribution mechanism

Distribution Mechanism

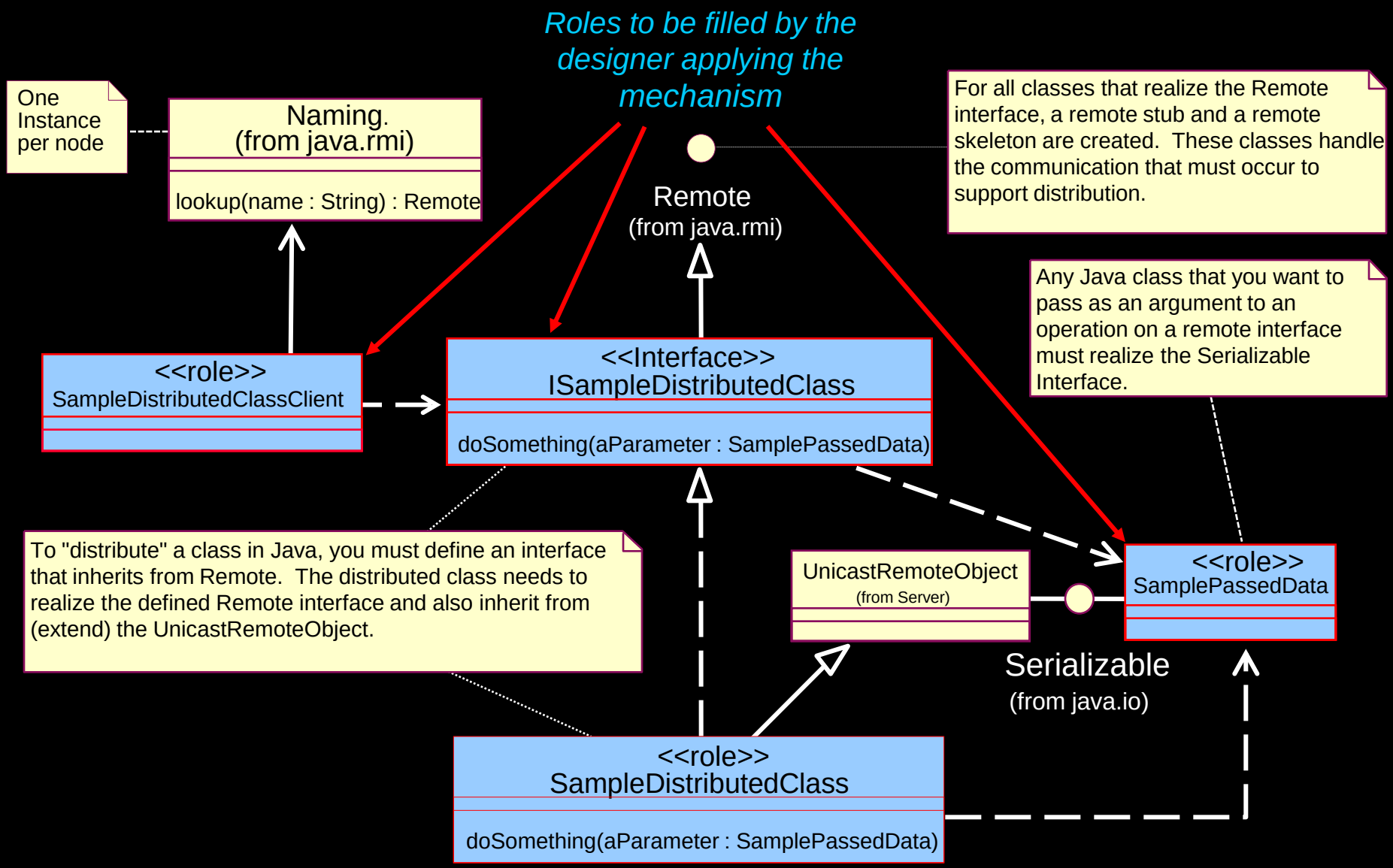
- RMI was chosen as the implementation mechanism for distribution



Design Mechanisms: Distribution: RMI

- Distribution characteristics
  - Latency
  - Synchronicity
  - Message Size
  - Protocol

Remote Method Invocation (RMI) (continued)



Example: Incorporating RMI