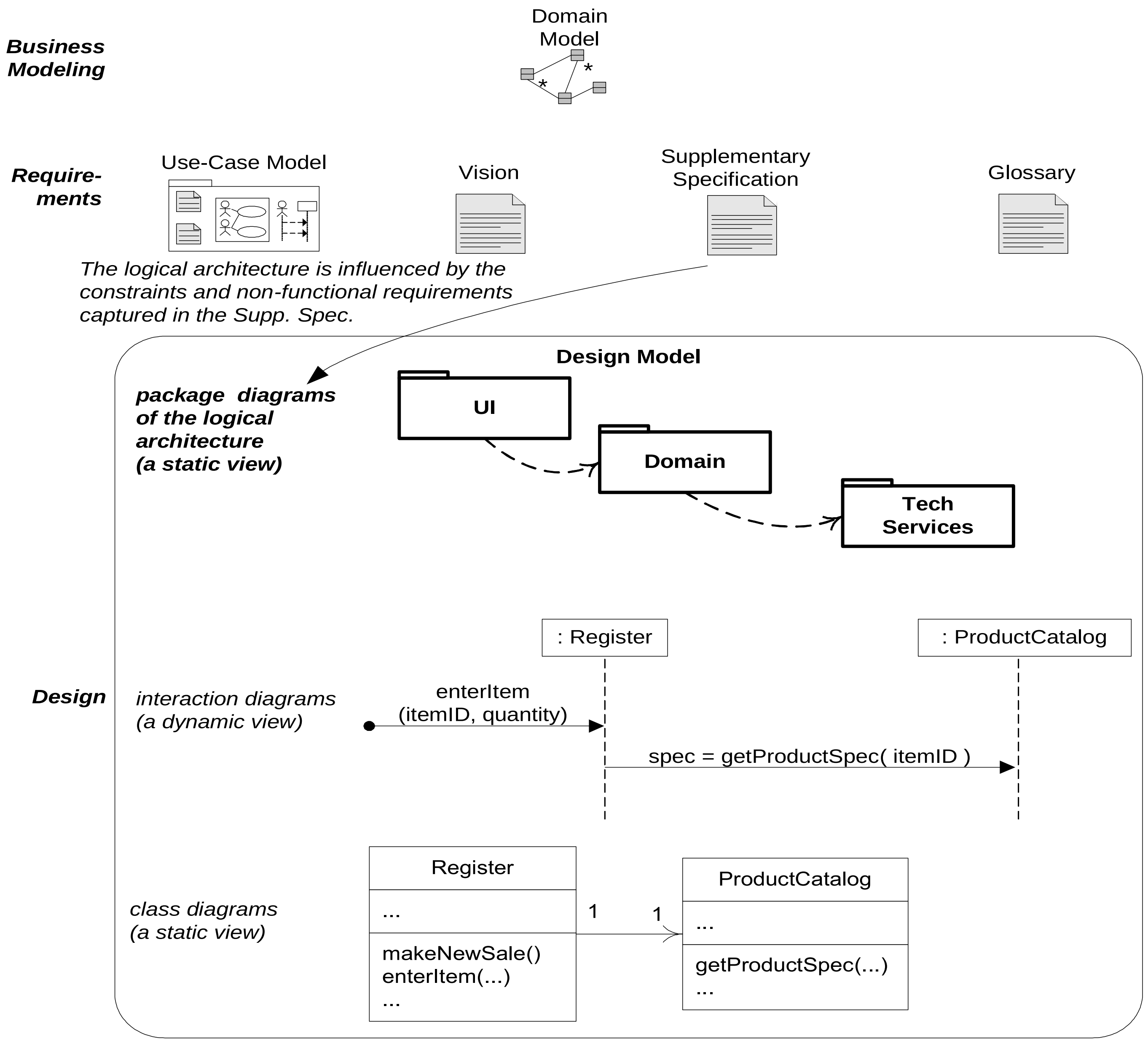
Logical Architecture and UML Package Diagrams

徐迎晓
xuyingxiao@126.com
http://javabook.126.com
复旦大学软件学院

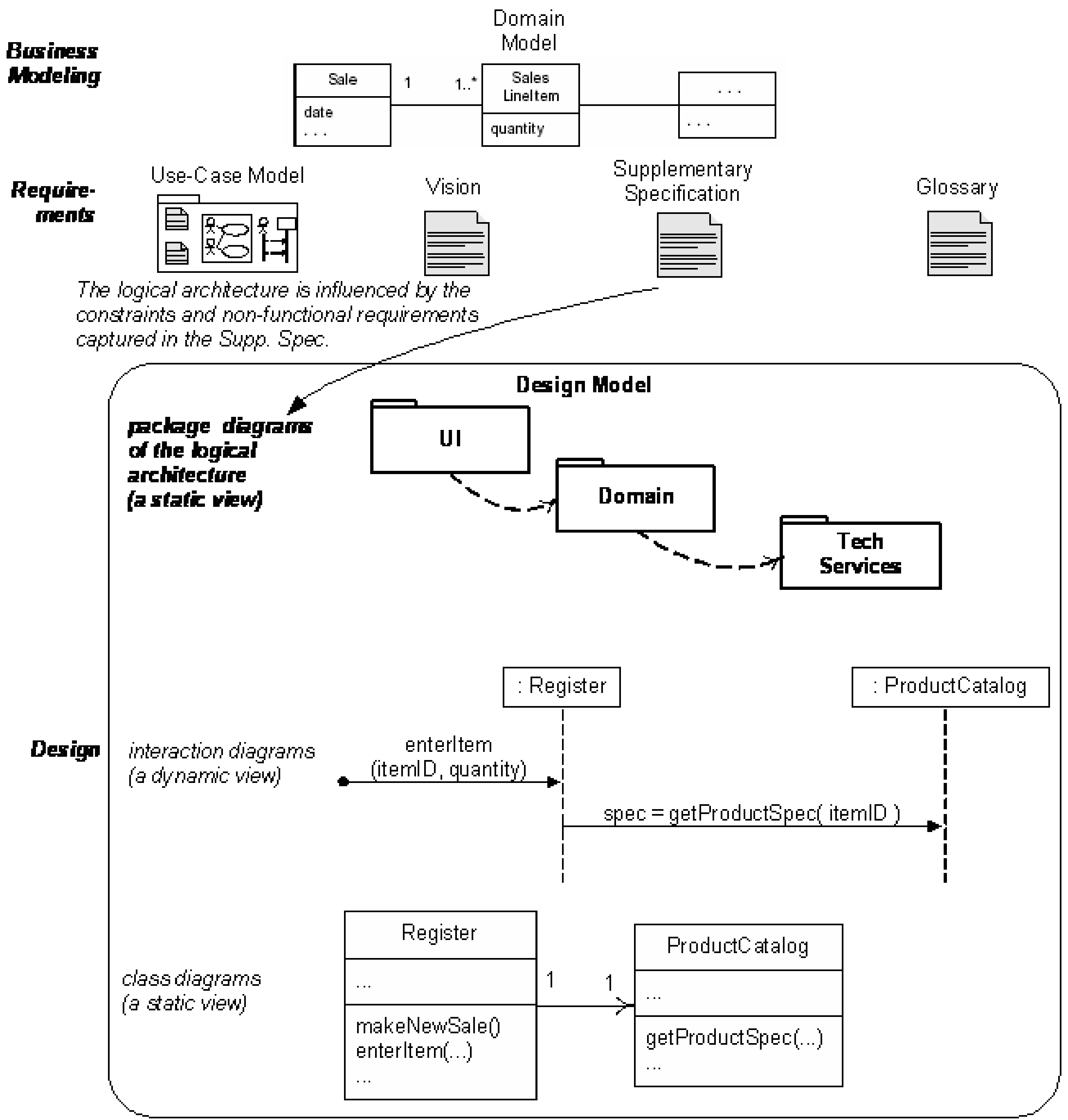
Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

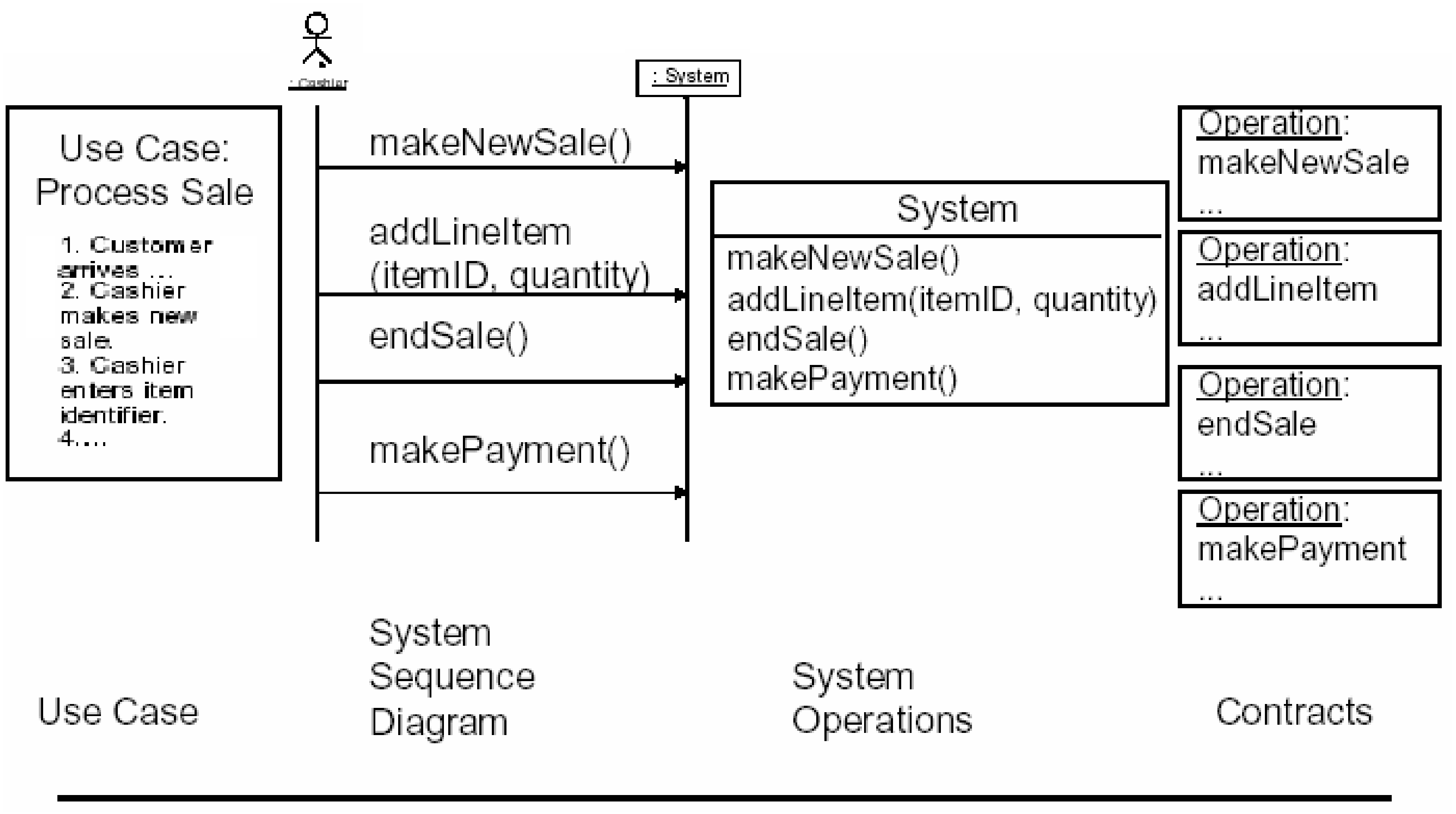
Sample UP Artifact Relationships


Business Modeling
Domain Model
*
*
Require-ments
Use-Case Model
Vision
Supplementary Specification
Glossary
The logical architecture is influenced by the constraints and non-functional requirements captured in the Supp. Spec.
Design Model
package diagrams of the logical architecture (a static view)
UI
Domain
Tech Services
Design
interaction diagrams (a dynamic view)
: Register
: ProductCatalog
enterItem (itemID, quantity)
spec = getProductSpec(itemID)
class diagrams (a static view)
Register
...
makeNewSale()
enterItem(...)
...
1
1
ProductCatalog
...
getProductSpec(...)
...

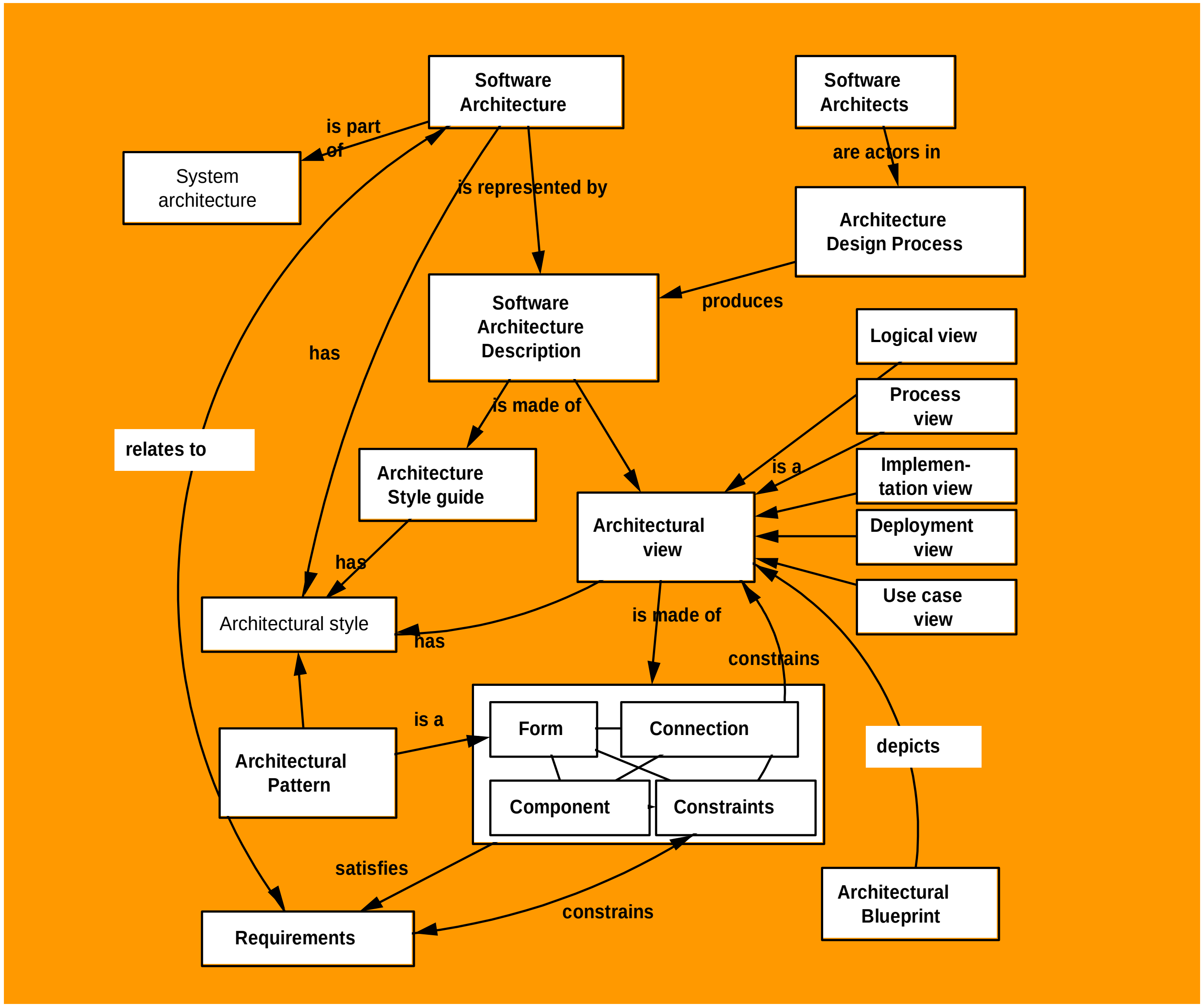
Sample UP Artifact Relationships



Business Modeling
Domain Model
Sale
date
. . .
1
1..*
Sales LineItem
quantity
. . .
. . .
Require-ments
Use-Case Model
Vision
Supplementary Specification
Glossary
The logical architecture is influenced by the constraints and non-functional requirements captured in the Supp. Spec.
Design Model
package diagrams of the logical architecture (a static view)
UI
Domain
Tech Services
Design
interaction diagrams (a dynamic view)
: Register
: ProductCatalog
enterItem (itemID, quantity)
spec = getProductSpec(itemID)
class diagrams (a static view)
Register
...
makeNewSale()
enterItem(...)
...
1
1
ProductCatalog
...
getProductSpec(...)
...


: Cashier
: System
Use Case: Process Sale
1. Customer arrives …
2. Cashier makes new sale.
3. Cashier enters item identifier.
4.…
makeNewSale()
addLineItem (itemID, quantity)
endSale()
makePayment()
System
makeNewSale()
addLineItem(itemID, quantity)
endSale()
makePayment()
Operation: makeNewSale …
Operation: addLineItem …
Operation: endSale …
Operation: makePayment …
Use Case
System Sequence Diagram
System Operations
Contracts

Architecture metamodel



Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

Definition

- Software Architecture:
- large scale-the Big Ideas in the forces, organization, styles, patterns, responsibilities, collaborations, connections and motivations of a system and major subsystems.

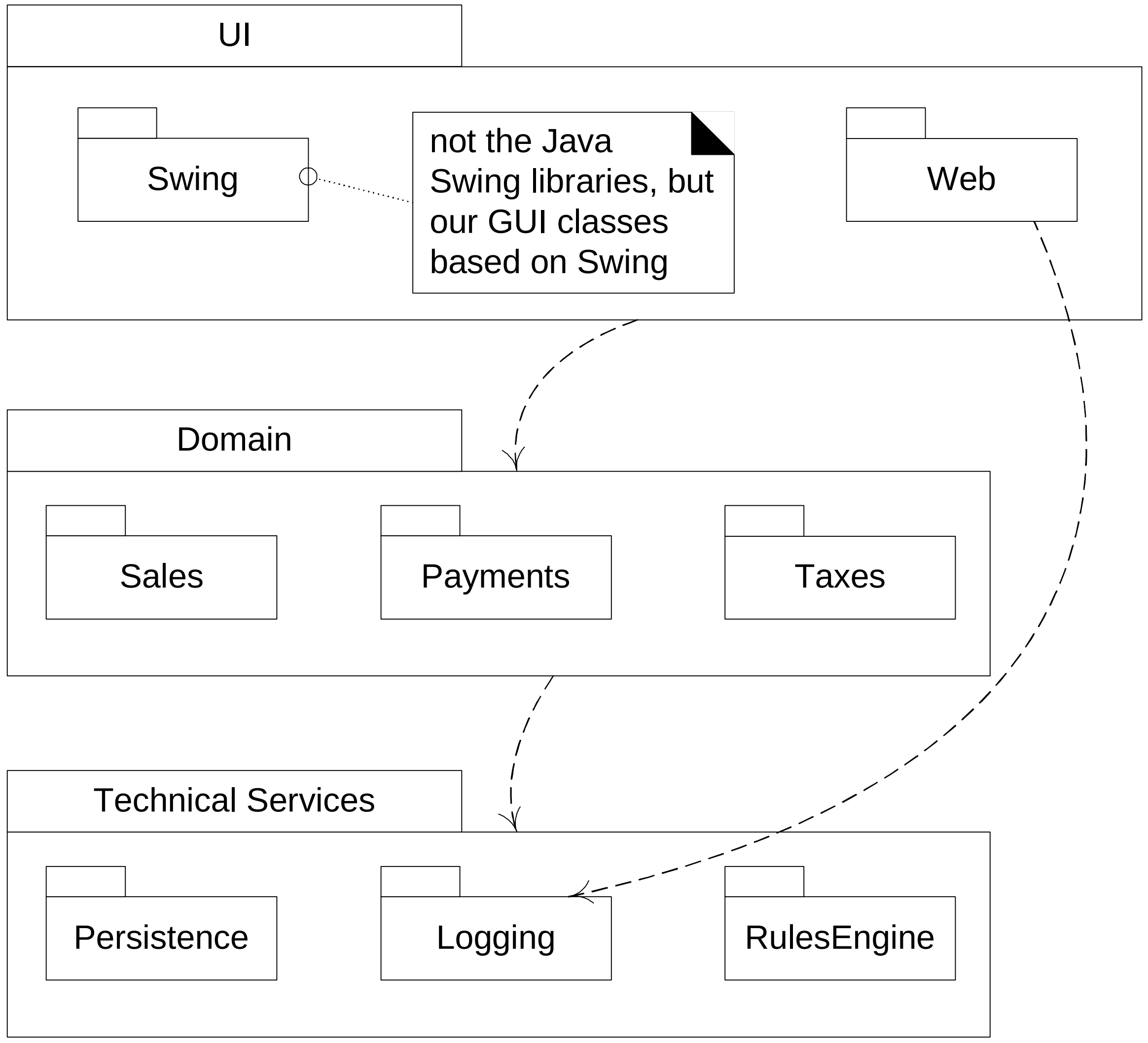
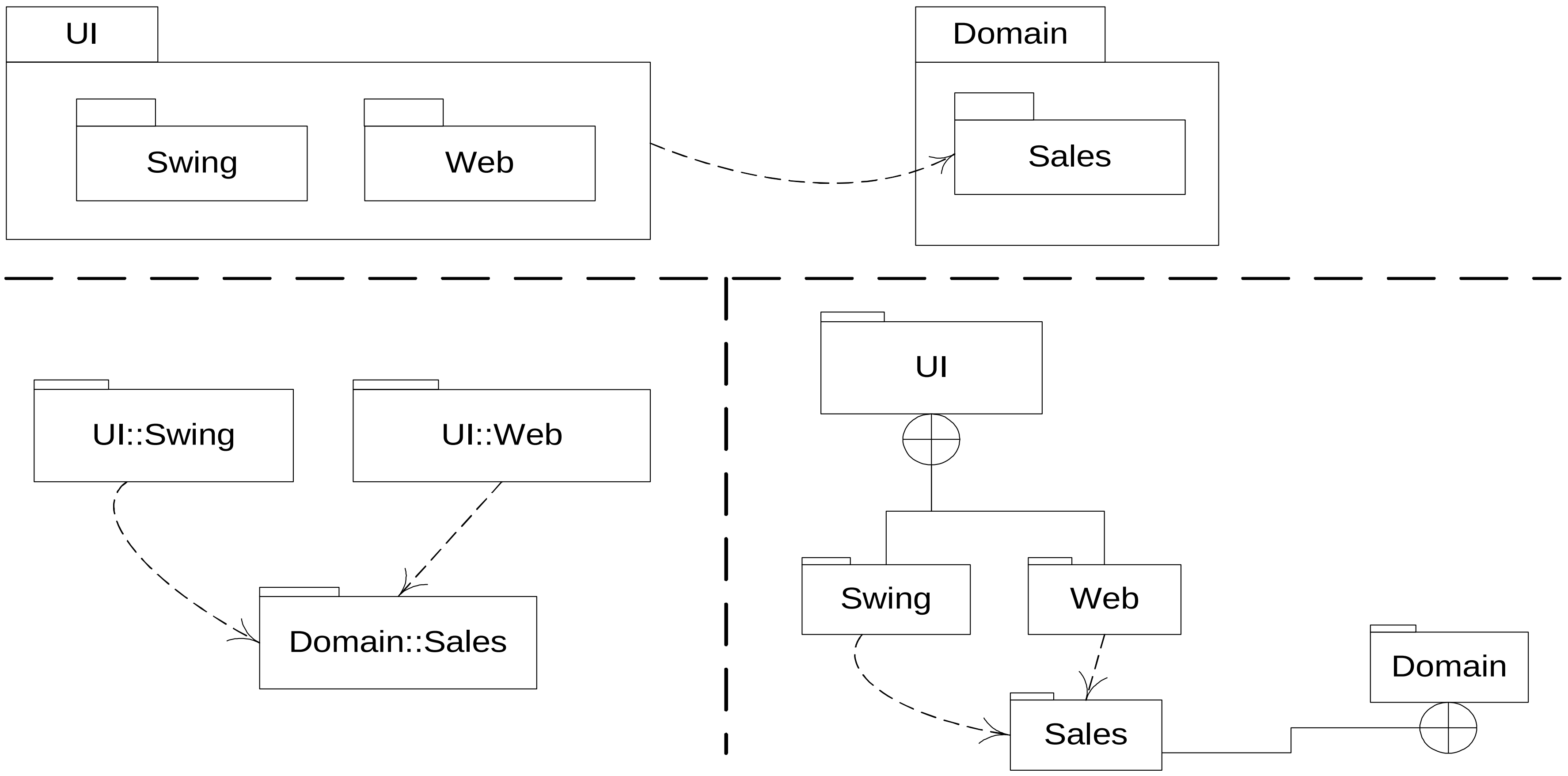

UI
Swing
not the Java Swing libraries, but our GUI classes based on Swing
Web
Domain
Sales
Payments
Taxes
Technical Services
Persistence
Logging
RulesEngine

Fig. 13.3


UI
Swing
Web
Domain
Sales
UI::Swing
UI::Web
Domain::Sales
UI
Swing
Web
Sales
Domain

Definition variance

- noun
  - the architecture includes the organization and structure of the major elements of the system.
- verb
  - architecture is part investigation and part design work.

Definition

- Architectural investigation: involves functional and non-functional requirements that have impact on system design.
  - Some of these are: Market trends, performance, cost and points of evolution.
- Architectural Design: is the resolution of these requirements in the design of software.

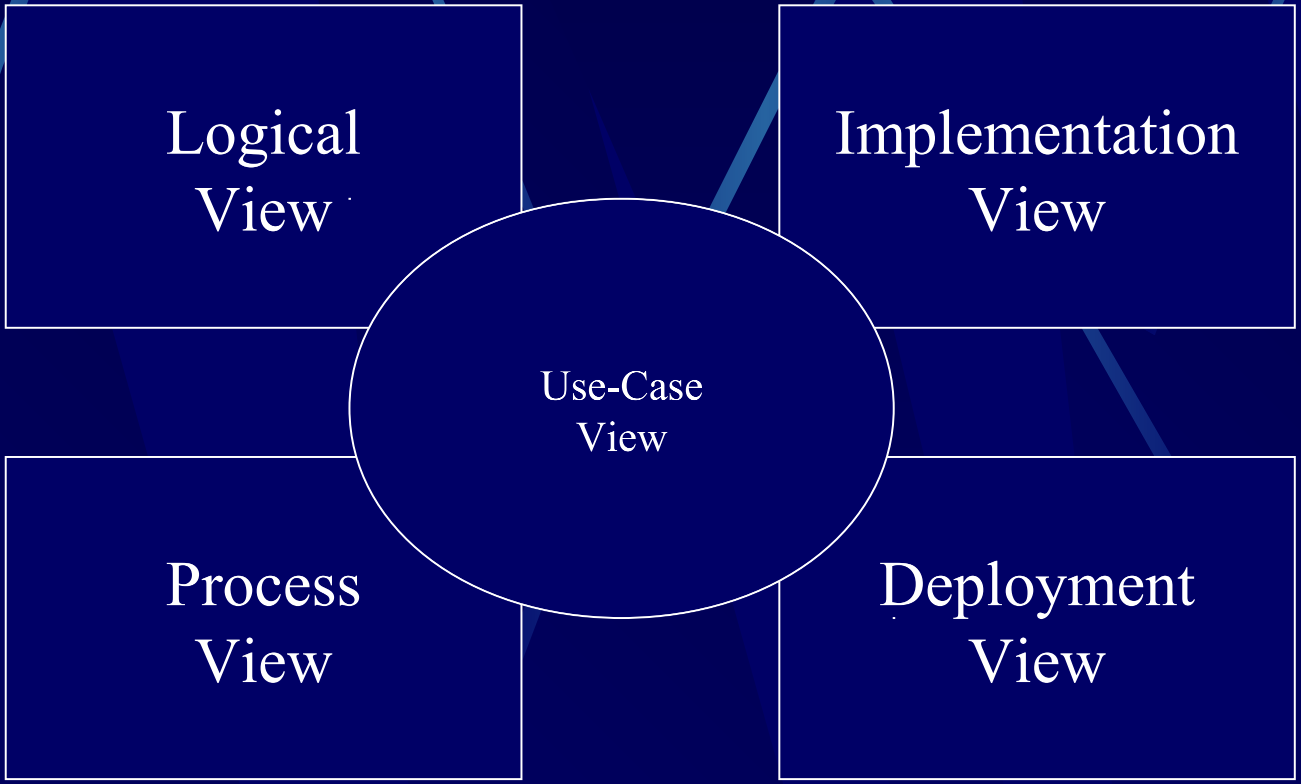
Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

Architectural Dimension and Views in UP

- The common dimensions are:
  - The logical architecture, describes the system in terms of its conceptual organization in layers, packages, classes, interfaces and subsystems.
  - The deployment architecture, describes the system in terms of the allocation of process to processing unit and network configurations.

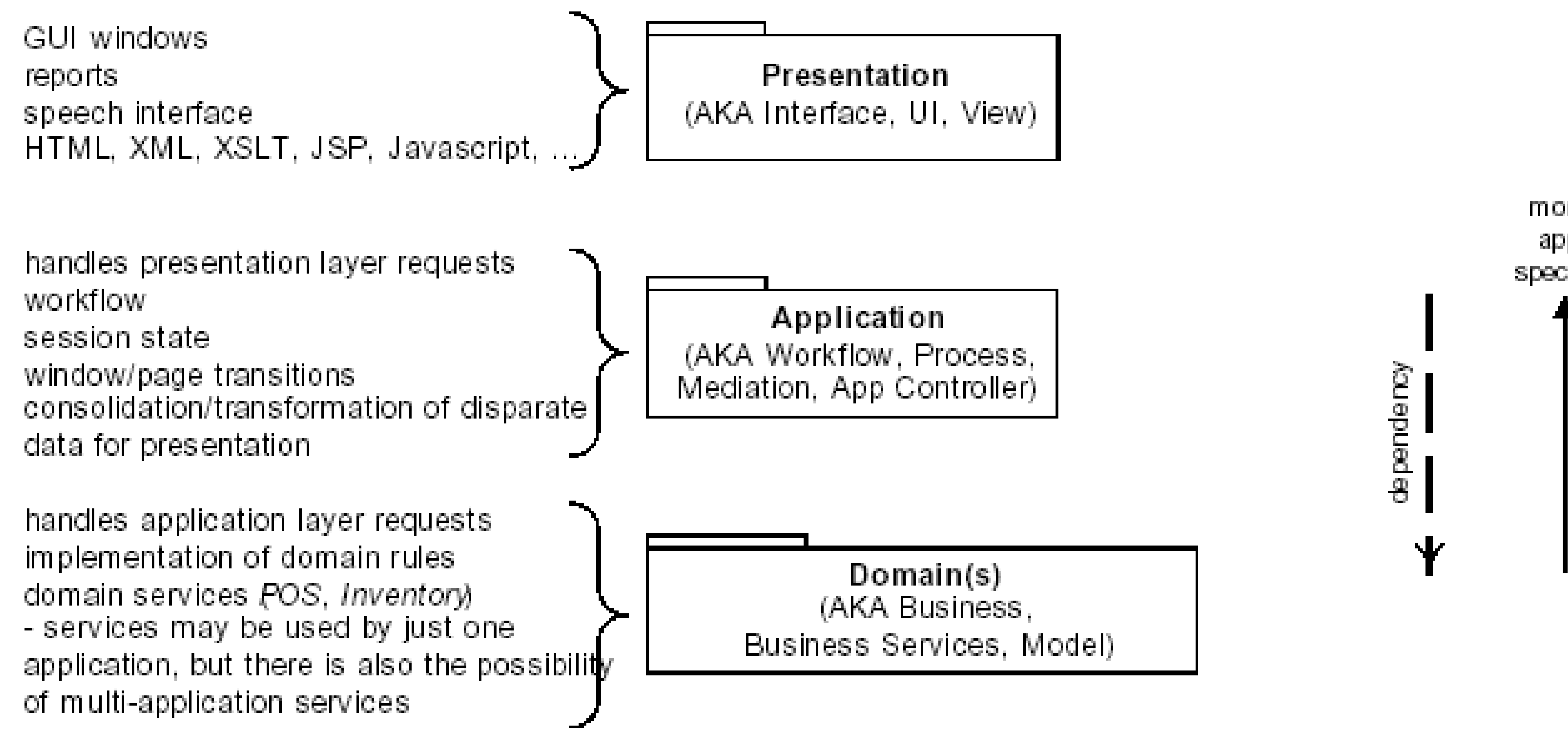
Architecture Dimensions and Views



Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

Architectural Pattern: Layers

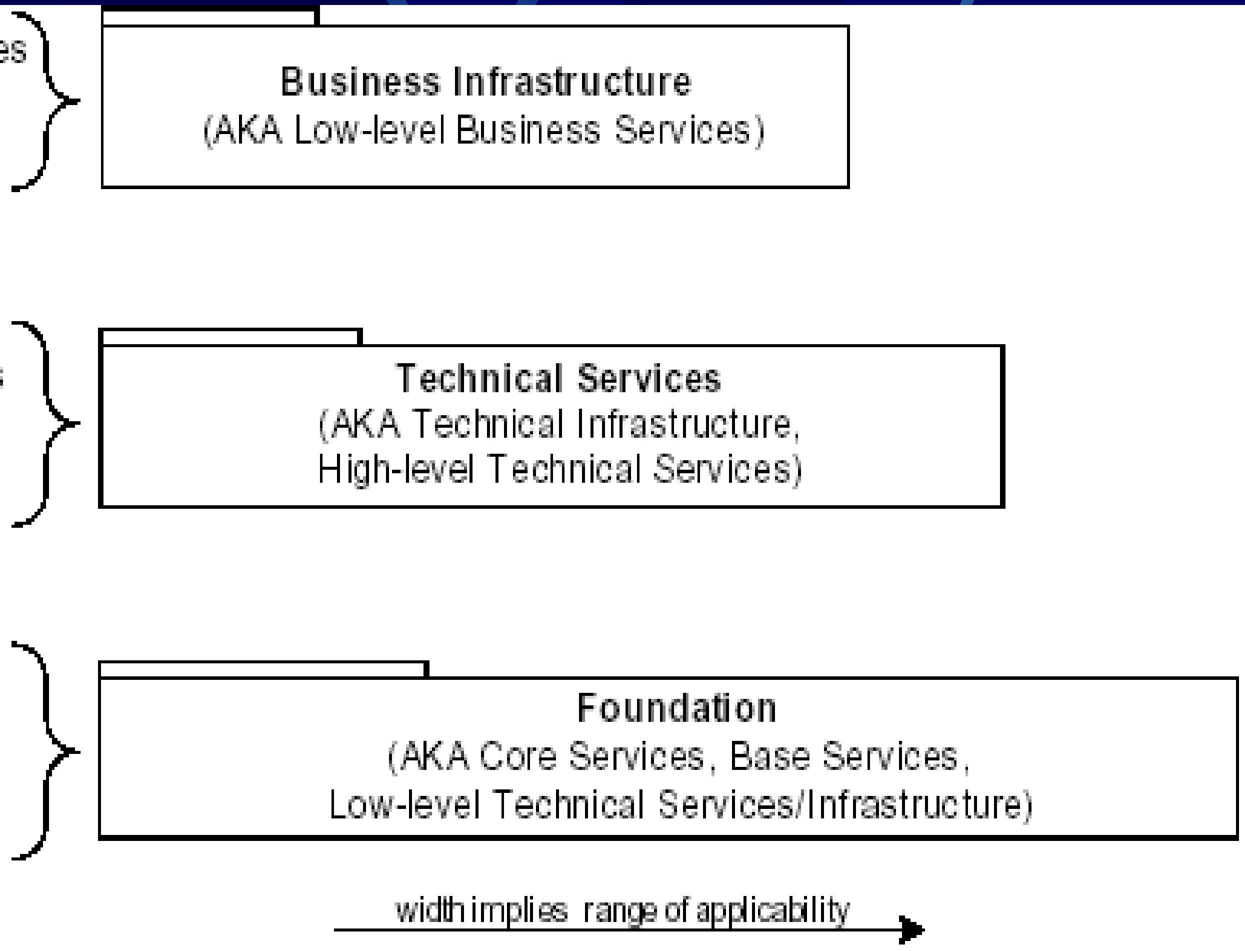


GUI windows
reports
speech interface
HTML, XML, XSLT, JSP, Javascript, ...

handles presentation layer requests
workflow
session state
window/page transitions
consolidation/transformation of disparate data for presentation

handles application layer requests
implementation of domain rules
domain services (*POS*, *Inventory*)
- services may be used by just one application, but there is also the possibility of multi-application services

Presentation
(AKA Interface, UI, View)
Application
(AKA Workflow, Process, Mediation, App Controller)
Domain(s)
(AKA Business, Business Services, Model)
dependency
mo
ap
spec


Business Infrastructure
(AKA Low-level Business Services)
Technical Services
(AKA Technical Infrastructure,
High-level Technical Services)
Foundation
(AKA Core Services, Base Services,
Low-level Technical Services/Infrastructure)
width implies range of applicability

very general low-level business services used in many business domains
CurrencyConverter

(relatively) high-level technical services and frameworks
Persistence, Security

low-level technical services, utilities, and frameworks
data structures, threads, math, file, DB, and network I/O

What is a layer?

- “A layer is a coarse grained grouping of classes packages or subsystems that has cohesive responsibility for a major aspect of the system.”
- Higher layers call upon the services of lower layers.

Architectural Patterns and Pattern Categories

- Architectural patterns: Relates to large-scale design and typically applied during the early iterations(in elaboration phase).
- Design patterns: Relates to small and medium-scale design of objects and frameworks.
- Idioms: Relates to language or implementation-oriented low-level design solutions.

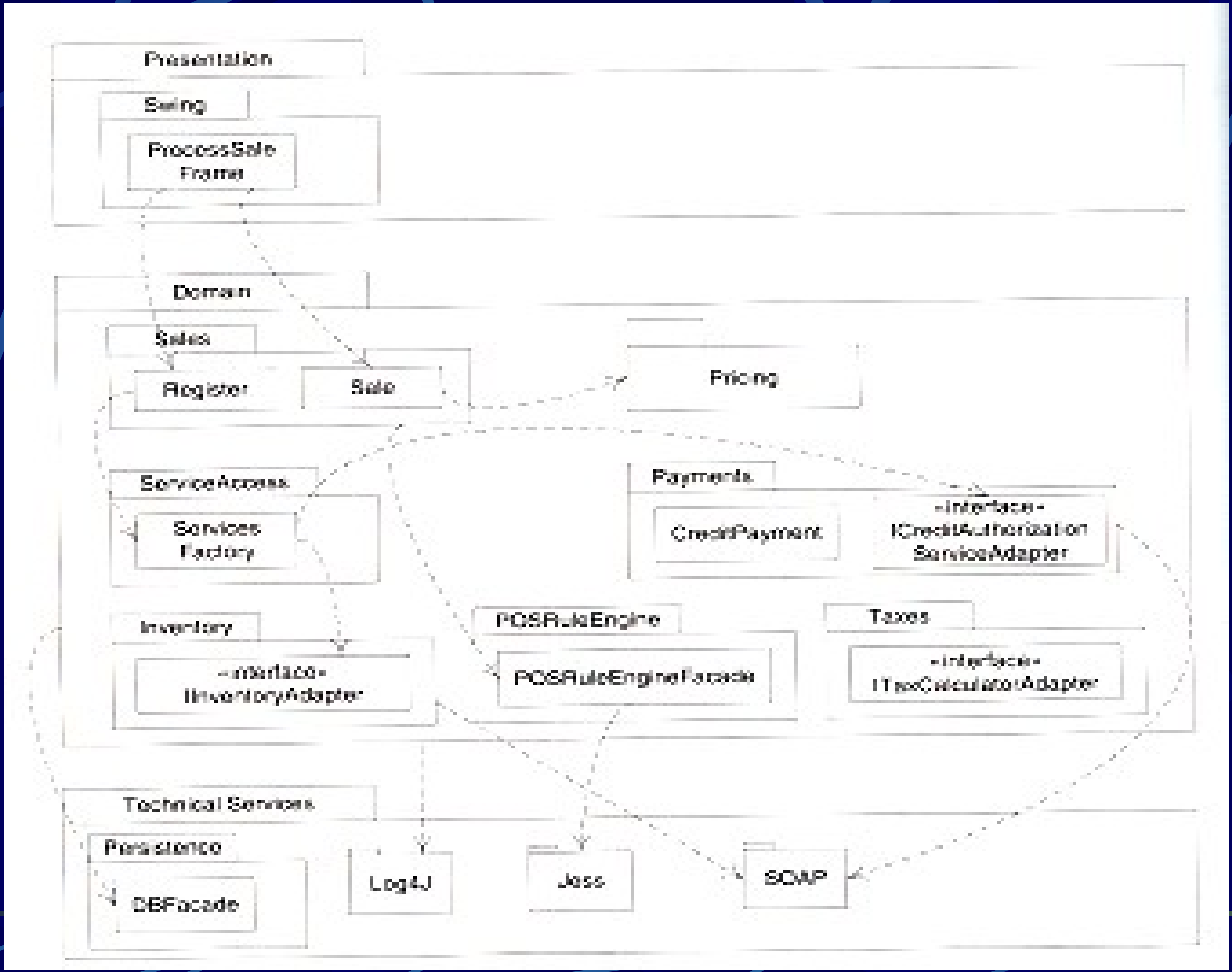
Architectural Pattern:Layers

- Idea behind Layer patterns:
  - Organize the large-scale logical structure of a system into discrete layers of distinct, related responsibilities with a clean, cohesive separation of concerns such that the “lower” layers are low-level and general services, and the higher layers are more application specific.
  - Collaboration and coupling is from higher to lower layers.

Inter-Layer and Inter-Package Coupling

- It is informative to include a diagram in the logical view that shows the coupling between the layers and packages.
- Following figure shows the coupling.

Partial coupling between Packages



Inter-Layer and Inter-Package Interaction

- Emphasizes the dynamics of how objects across the layers connect and communicate.
- The interaction diagram focuses on the logical view and on the collaborations between the layers and package boundaries.

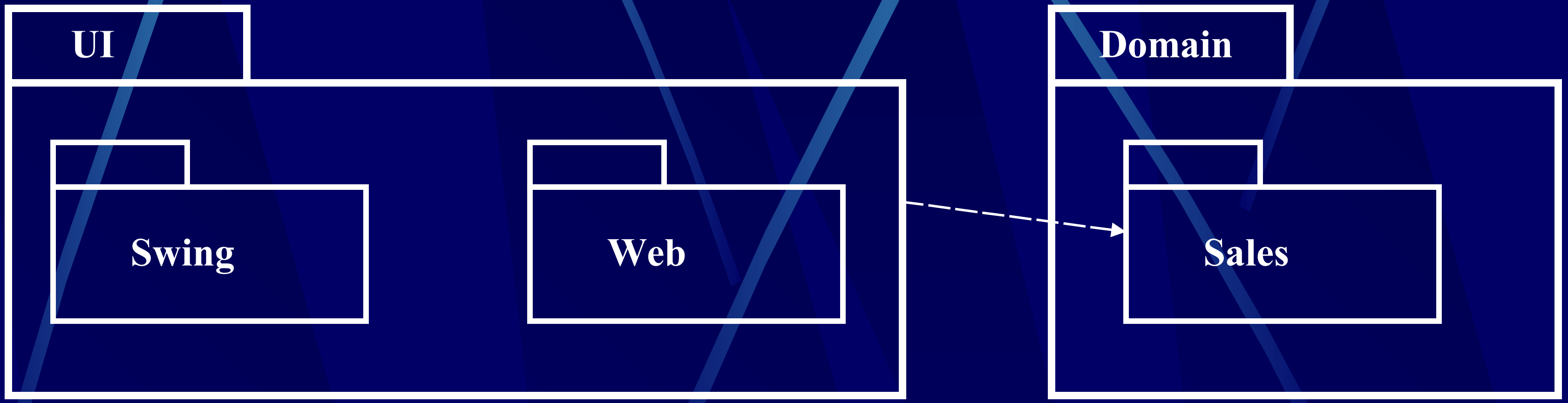
Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

Package Diagrams

- UML Package Diagrams are often used to show the contents of components, which are often packages in the Java sense.
- Each package represents a namespace.
- Packages, as components, can be nested inside other packages.

Package Diagram



Outline

Logical vs. Process and Deployment of Architecture

- Architectural Layers are a logical view of the architecture
- They are not a deployment view of elements to process.
- Depending on platform, all layers could be deployed within the same process on same node.
- Or across many computers.

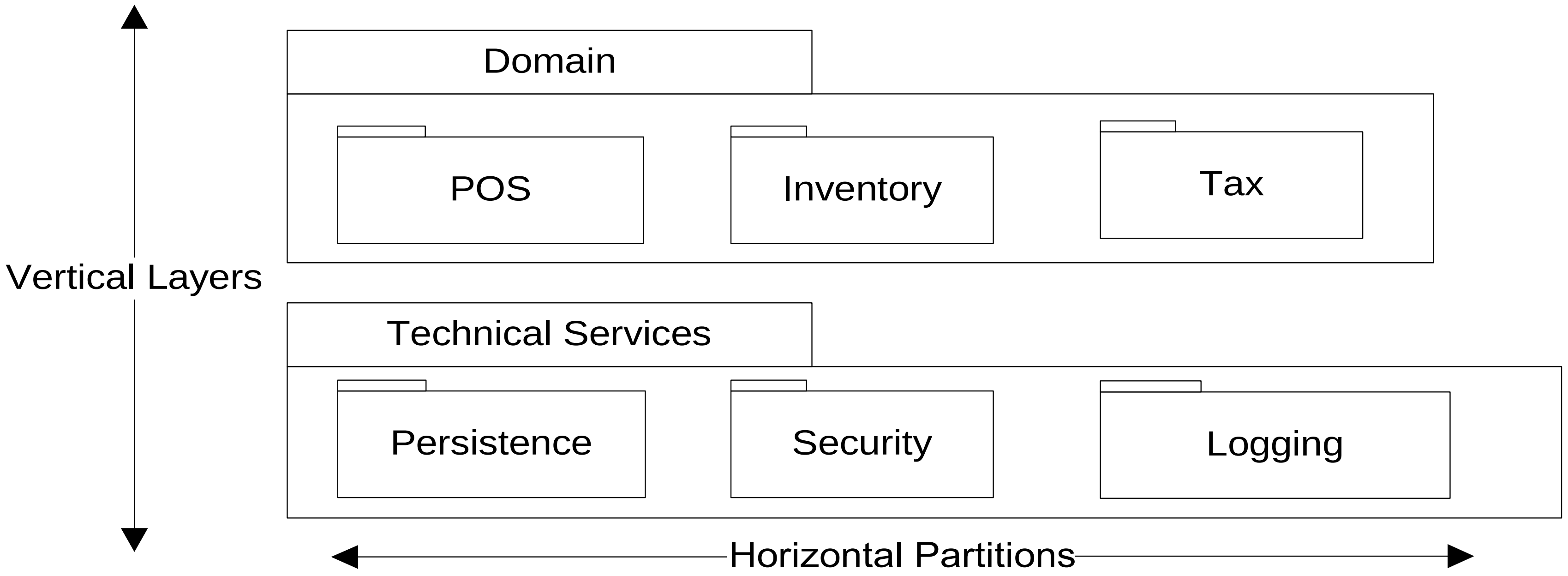
Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

Terminology:Tier, Layers, and Partitions

- Tier relates to physical processing node or clusters of node, such as “client tier”.
- Layers of an architecture represent the vertical slices
- Partitions represents a horizontal division of relatively parallel subsystems of a layer.

Fig. 13.6 Layers and Partitions



Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

How do we design application logic with objects?

- We could create one class and put all logic in it, but that violates the whole spirit of object orientation.
- We create software objects with names drawn from the real world, and assign application logic responsibilities to them.
- It takes a lot of skill and experience to do a good job of choosing objects and assigning responsibilities.

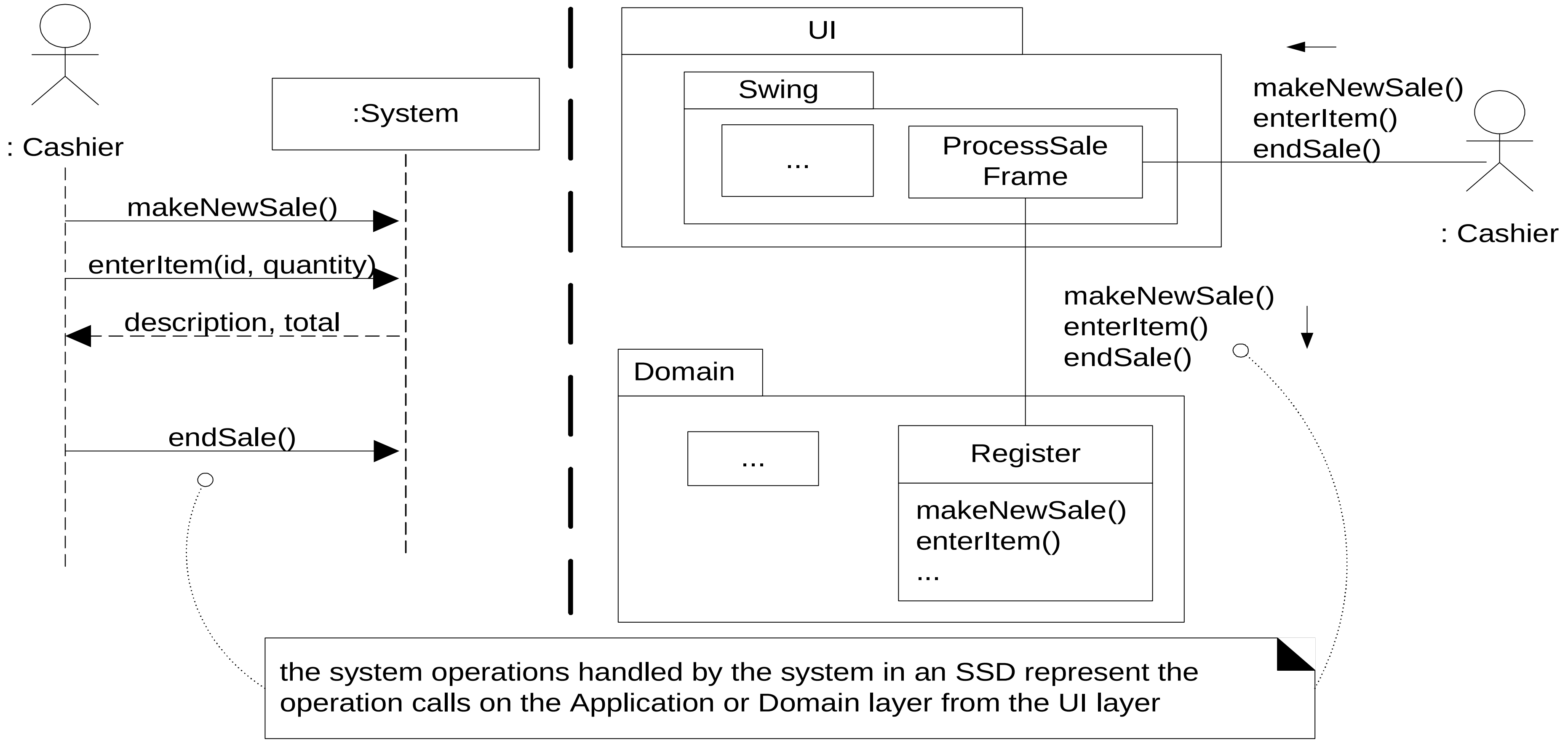
Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

Domain Layer and Domain Model

- These are not the same thing. Domain model shows the real world, while the Domain layer shows the software architecture.
- But the Domain model inspires the Domain layer, and is the source of many of the concept, especially class names.
- Do not confuse the problem with the solution.

Fig. 13.8



the system operations handled by the system in an SSD represent the operation calls on the Application or Domain layer from the UI layer

Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

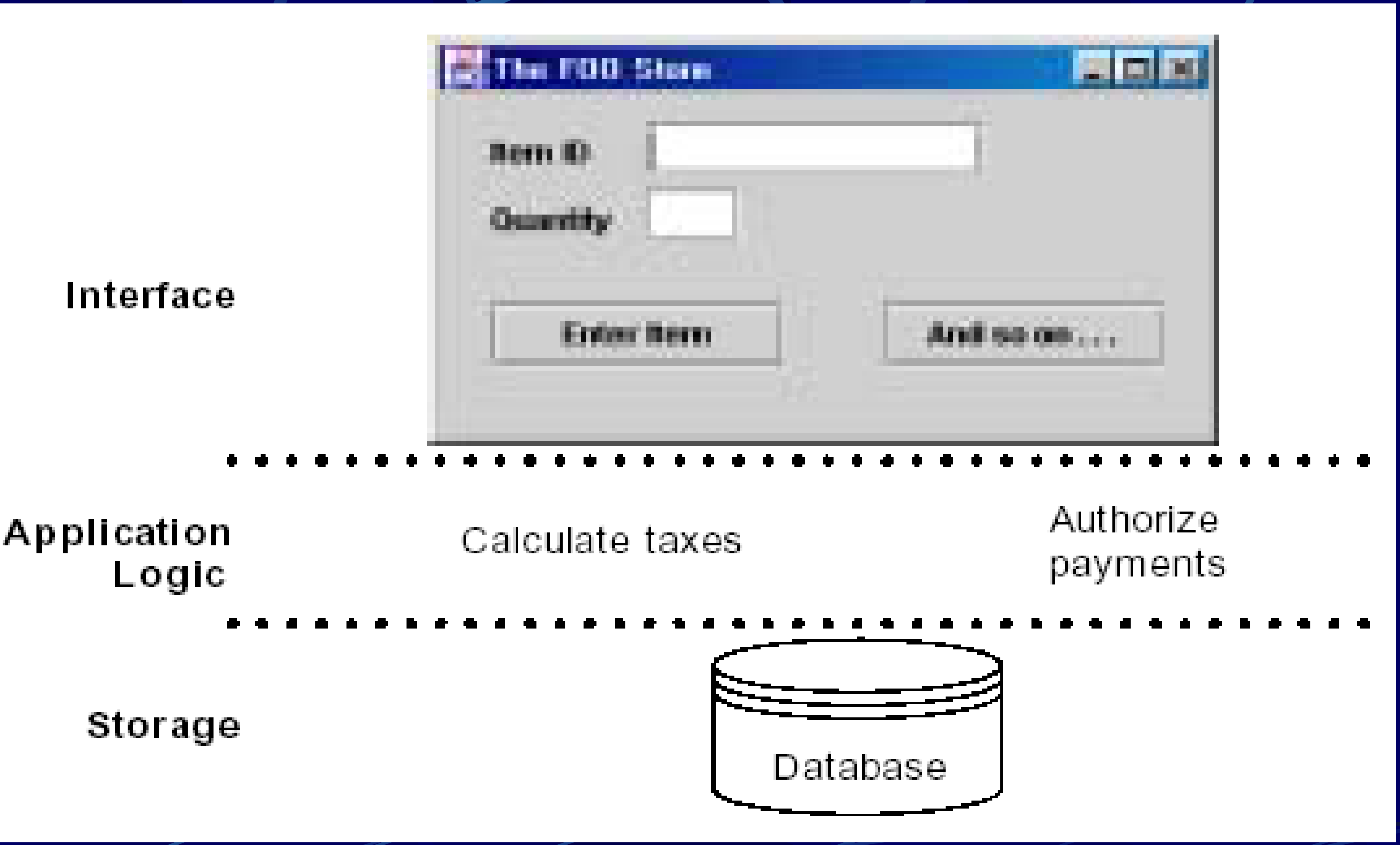
Information Systems

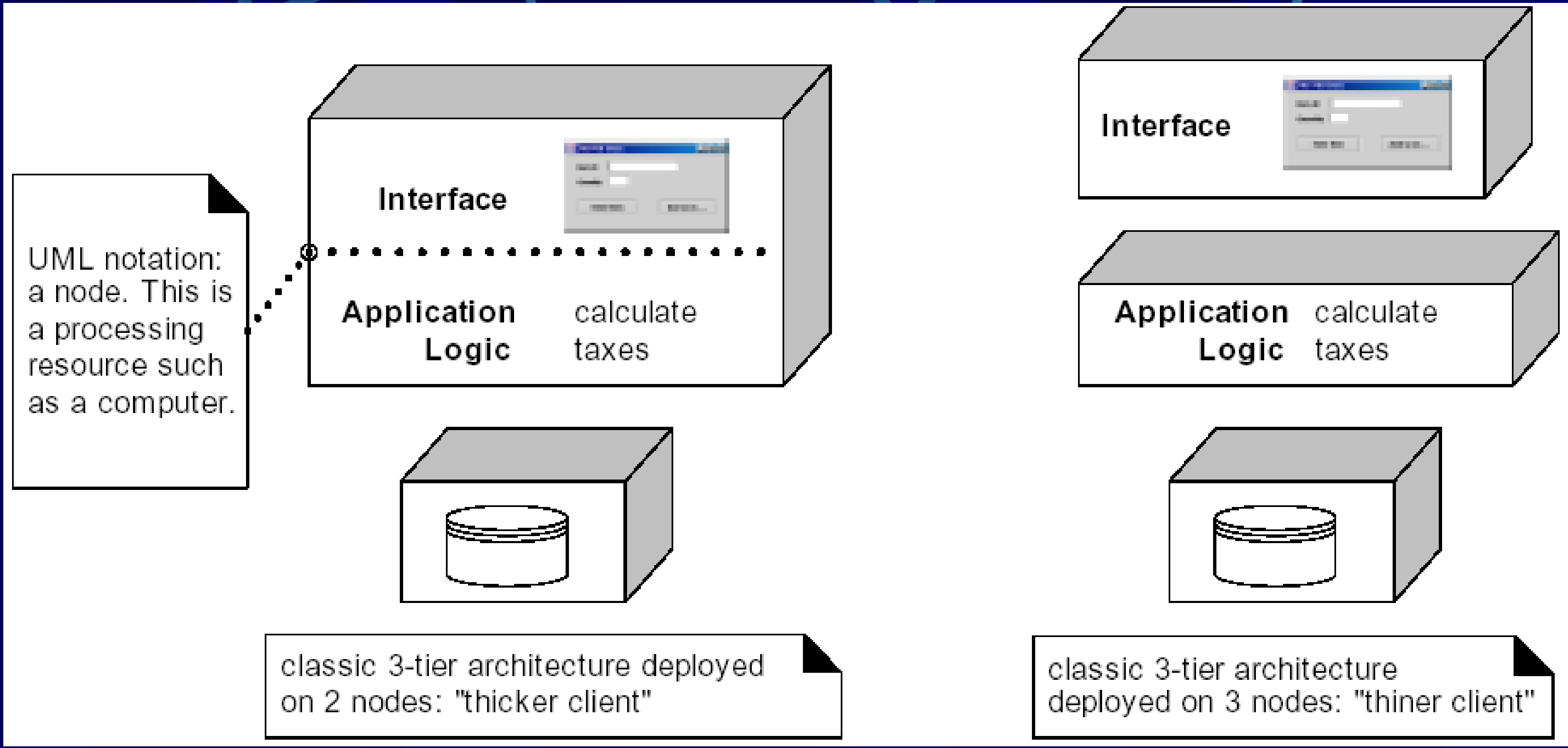
- In IS layered architecture was known as three-tier architecture.
- A three-tier architecture has interface, Application logic and a storage.
- The singular quality of 3-tier architecture is:
  - Separation of the application logic into distinct logical middle tier of software.
  - The interface tier is relatively free of application processing.

Information Systems(cont..)

- The middle tier communicates with the back-end storage layer.
- The following is an example of 3-tier architecture.

Example:


The FOO Store
Item ID
Quantity
Enter Item
And so on . . .
Interface
Application Logic
Calculate taxes
Authorize payments
Storage
Database


UML notation: a node. This is a processing resource such as a computer.
Interface
Application Logic
calculate taxes
classic 3-tier architecture deployed on 2 nodes: "thicker client"
Interface
Application Logic
calculate taxes
classic 3-tier architecture deployed on 3 nodes: "thiner client"

Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

Two-tier Design

- In this design, the application logic is placed within window definitions, which read and writes directly to database.
- There is no middle tier that separates out the application logic.

Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

The Model-View Separation Principle

- The principle states that model(domain) objects should not have direct knowledge of view(presentation) objects.
- Furthermore, the domain classes should encapsulate the information and behavior related to application logic.

Outline

- Definition
- Architectural Dimension and Views
- Architectural Pattern: Layers
- Package Diagrams
- Logical vs. Process and Deployment of Architecture
- Terminology:Tier, Layers, and Partitions
- How do we design application logic with objects?
- Domain Layer and Domain Model
- Information Systems
- Two-tier Design
- The Model-View Separation Principle
- Need for Model-View separation

Need for Model-View separation

- To support cohesive model definitions that focus on the domain process, rather than on interfaces.
- To allow separate development of the model and user interface layers.
- To minimize the impact of requirements changes in the interface upon the domain layer.
- To allow new views to be easily connected to an existing domain layer, without affecting the domain layer.

Continue..

- To allow multiple simultaneous views on the same model object.
- To allow execution of the model layer independent of the user interface layer
- To allow easy porting of the model layer to another user interface framework.