Documenting Software Architectures

Incorrect documentation is often worse than no documentation.
Bertrand Meyer

LECTURE 09

DR USMAN NASIR

Documenting an Architecture

Documenting an Architecture

- The architecture need to be recorded for stakeholders.
  - Documents needs to be created and updated so they remain relevant.
  - Should be detailed, updated and organized for future reference.

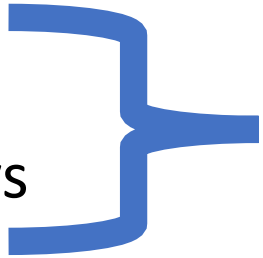
- Remember an architect
  - can not answer every question and explain everything about his decision.
  - create software architecture and leave the job.
  - needs to produce good documentation for future re-work.
  - helps record reason about the architecture design.

- The documentation is focused towards the audience, **key stakeholders of the project/product**

- Architecture documentation has four main uses:
  - Educating stakeholders
  - Communication among stakeholders
  - Basis for system analysis and construction
  - Basis for forensics when an incident occurs

Information types in architecture documentation

- Module views
- Component-and-connector views
- Allocation views
- Behavioural aspects of the system
- Software Interfaces



Focus of our Assignment 2

- How many views should be documented?
  - Different views support different goals and uses.
    - The views you should document depend on the uses you expect to make of the documentation.
  - *For a **larger system** at least one module view, one C&C view and one allocation view should be in architecture document*
  - Each view has a cost and a benefit; you should ensure that the benefits of maintaining a view outweigh its costs.

Documenting Behaviour

- Documenting behavioural aspects of an architecture provides benefits during development and maintenance of system.

- Behaviour documentation
  - helps to gain understanding of a system
  - can identify potential deadlocks and bottlenecks
  - clarifies to developers the steps and states involved in the operations.

- Behaviour documentation complements structural views by describing how architecture elements interact via their structures.

Documenting Interfaces

- Documenting interfaces is an important part of documenting architecture.
  - An interface is a boundary across which two independent entities meet and interact or communicate with each other.
- Documenting an interface consists of naming and identifying it and documenting it using an interface specification.

Seven Rules for Sound Documentation

- Rule 1: Write Documentation from the Reader's Point of View
- Rule 2: Avoid Unnecessary Repetition
- Rule 3: Avoid Ambiguity
  - Rule 3a: Explain Your Notation
- Rule 4: Use a Standard Organization scheme (document template)
- Rule 5: Record Rationale
- Rule 6: Keep Documentation Current but Not Too Current
- Rule 7: Review Documentation for Fitness of Purpose

Documentation Package

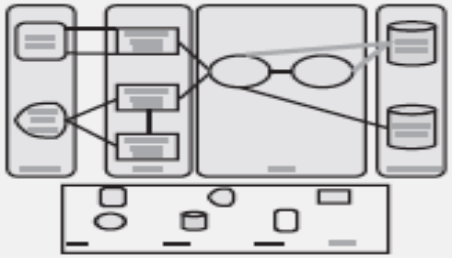
Documentation Package

- Documentation package consists of
  - Views
  - Documentation beyond views

View template

Template for a View

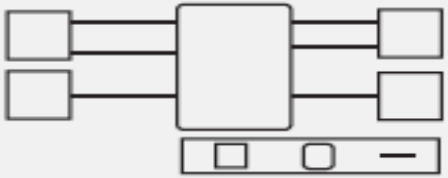
Section 1. Primary Presentation



Section 2. Element Catalog

- Section 2.A. Elements and Their Properties
- Section 2.B. Relations and Their Properties
- Section 2.C. Element Interfaces
- Section 2.D. Element Behavior

Section 3. Context Diagram



Section 4. Variability Guide

Section 5. Rationale

Documenting a View

- **Section 1: The Primary Presentation.**
  - The *primary presentation* shows the elements and relations of the view.
  - The primary presentation should contain the information you wish to convey about the system—in the vocabulary of that view.
  - The primary presentation is often graphical (e.g., UML diagram)
    - If your primary presentation is graphical, **make sure to include a key that explains the notation.**
    - Lack of a key is the most common mistake that we see in documentation in practice or wrong usage of the UML diagram
  - Occasionally the primary presentation will be textual, such as a table or a list.

Section 2: The Element Catalog.

- The *element catalog* details at least those elements depicted in the primary presentation.
  - For instance, if a diagram shows elements A, B, and C, then the element catalog needs to explain what A, B, and C are.
- Parts of the catalog:
  - *2A Elements and their properties*
  - *2B Relations and their properties*
  - *2C Element interfaces*
  - *2D Element behavior*

- **Section 3: Context Diagram.**
  - A *context diagram* shows how the system or portion of the system shown in this view relates to the systems' environment (humans, other systems, sensors, devices etc.)

- **Section 4: Variability Guide.**
  - A *variability guide* shows how to exercise any variation points that are a part of the architecture shown in this view. **Variation points are places in architecture where specific instances of flexibility have been built in**

Section 5: Rationale.

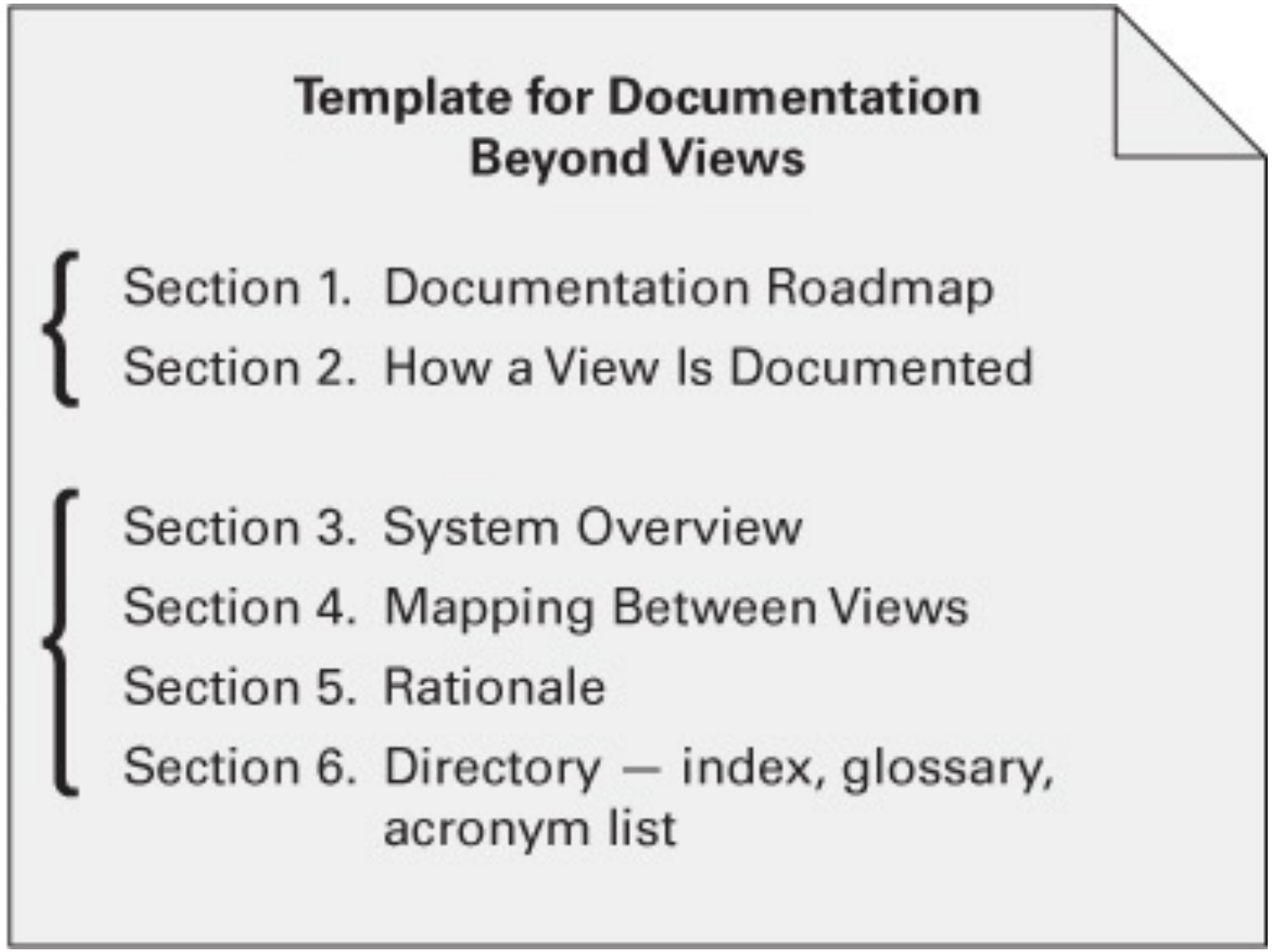
- The goal of this section is to explain why the design is as it is and to provide a convincing argument that it is sound.

- The choice of a pattern in this view should be justified here by describing the architectural problem that the chosen pattern solves and the rationale for choosing it over another.

Documentation beyond views

- Besides views additional information is documented. Documentation beyond views can be divided into two parts:
  - Overview of the architecture documentation.
    - This section explains how document is organized.
  - Information about the architecture.
    - purpose of the system, how views relate to one views, rationale behind design approaches, glossary etc.

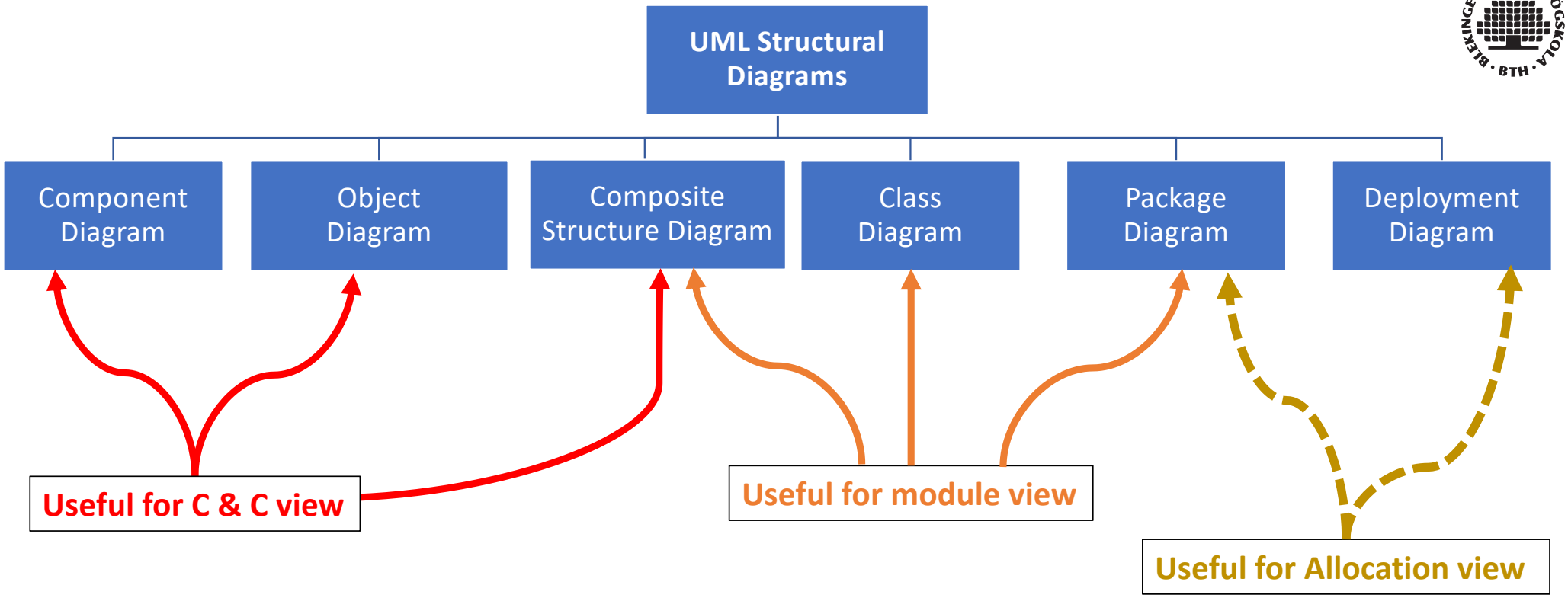
- To completely document a software architecture, you also need:
  - Development Roadmap
  - System overview/introduction
  - Rationale/explanation of why significant choices were made
  - References to relevant supporting documentation


Template for Documentation Beyond Views
Architecture documentation information
Section 1. Documentation Roadmap
Section 2. How a View Is Documented
Architecture information
Section 3. System Overview
Section 4. Mapping Between Views
Section 5. Rationale
Section 6. Directory — index, glossary, acronym list

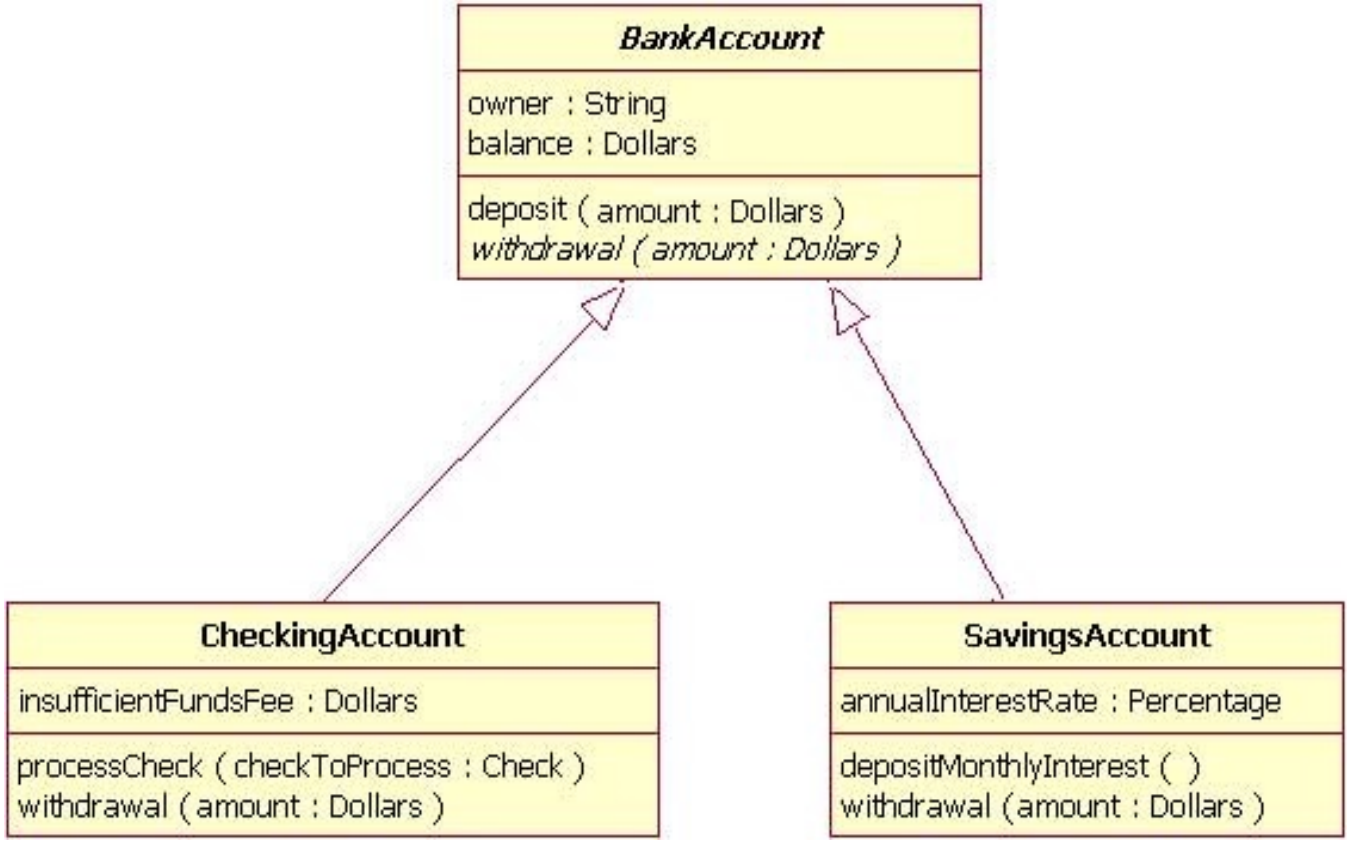
Documenting Views

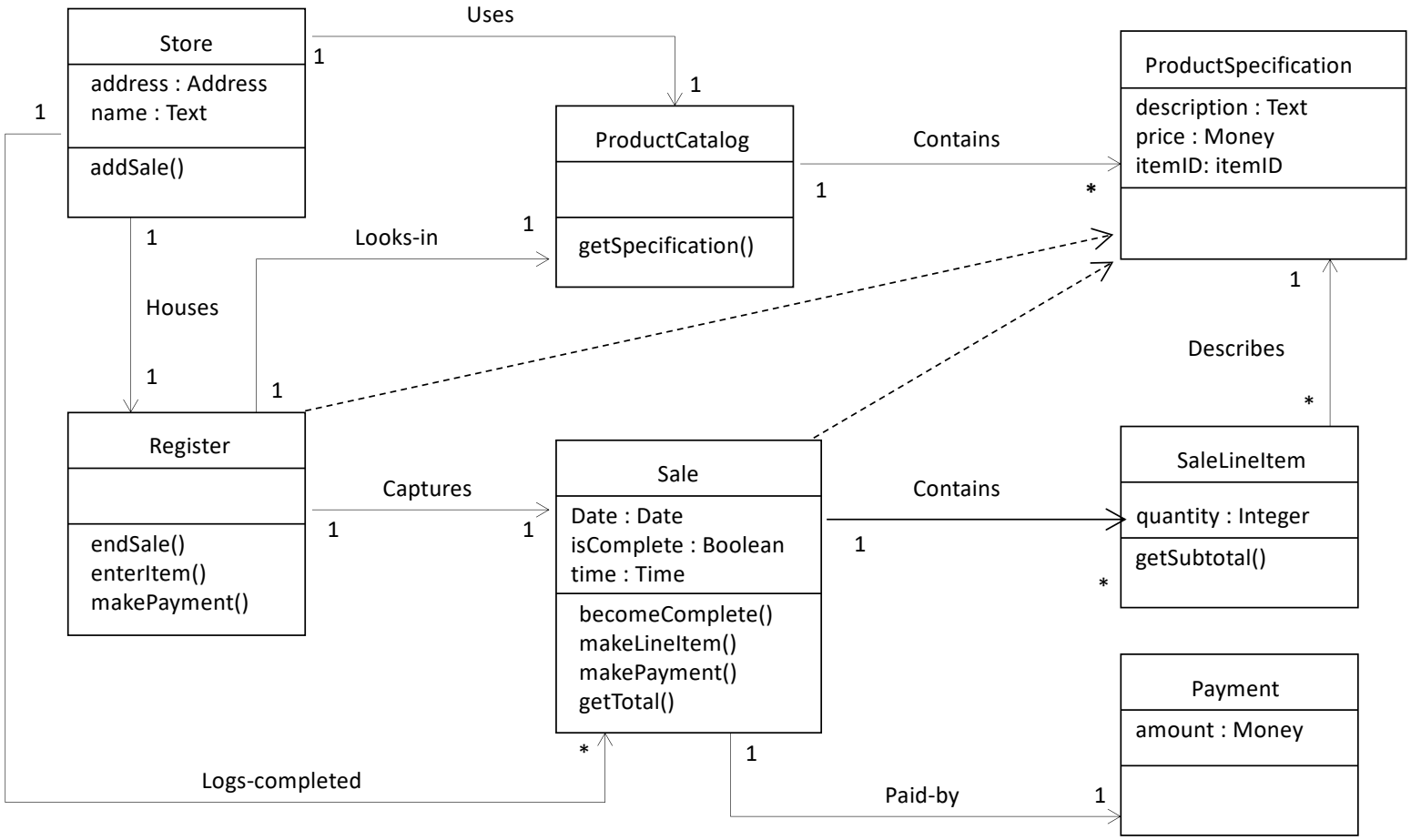
UML

- Unified Modeling Language (UML) is standardized visual language for modeling software designs
  - de-facto and de-jure standard
- Types of Diagrams in UML
  - Structural Diagrams – focus on static aspects of the software system
    - Class, Package, Object, Component, Deployment
  - Behavioural Diagrams – focus on dynamic aspects of the software system
    - Use-case, Interaction, State Chart, Activity

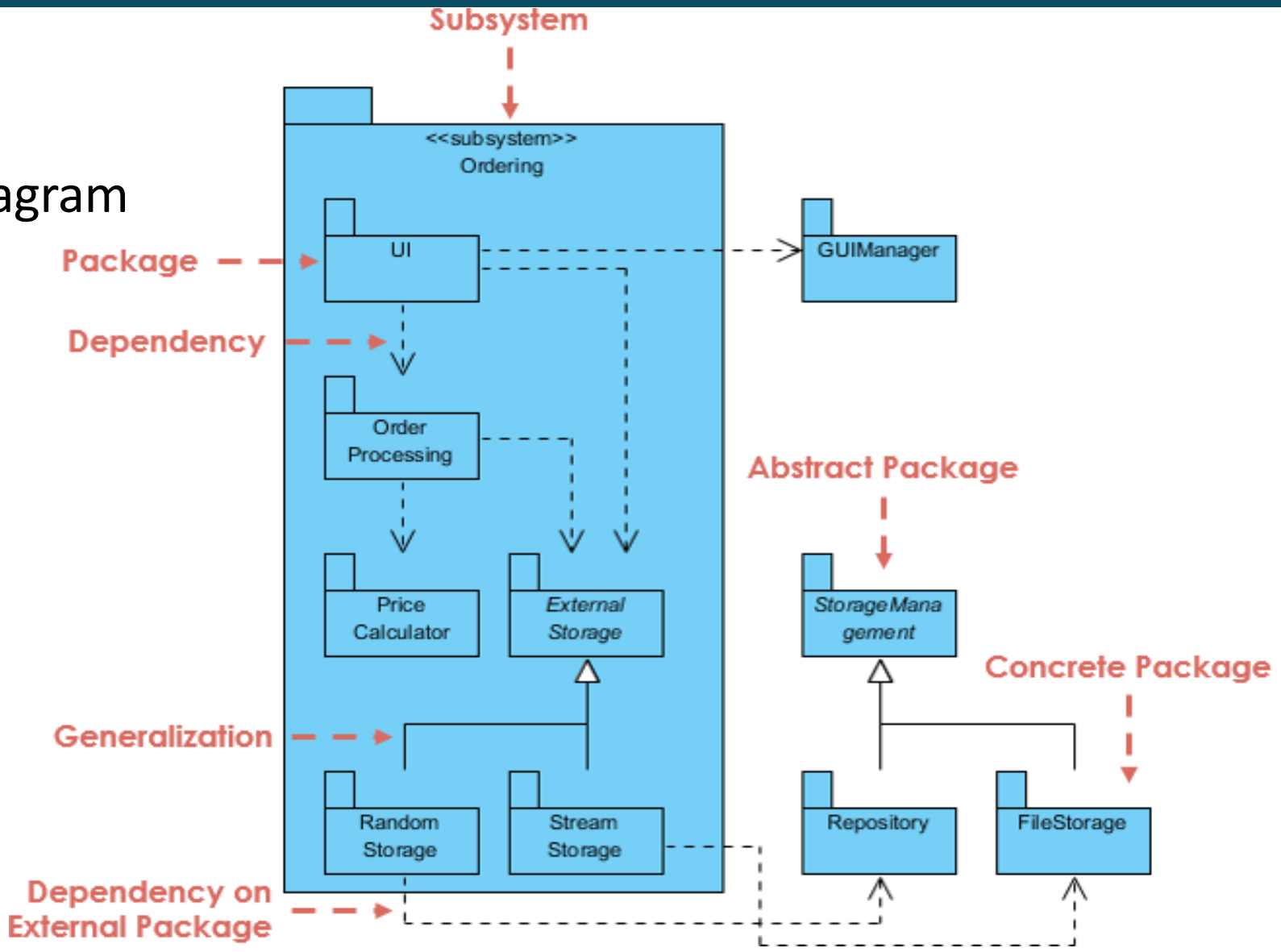

UML Structural Diagrams
Component Diagram
Object Diagram
Composite Structure Diagram
Class Diagram
Package Diagram
Deployment Diagram
Useful for C & C view
Useful for module view
Useful for Allocation view

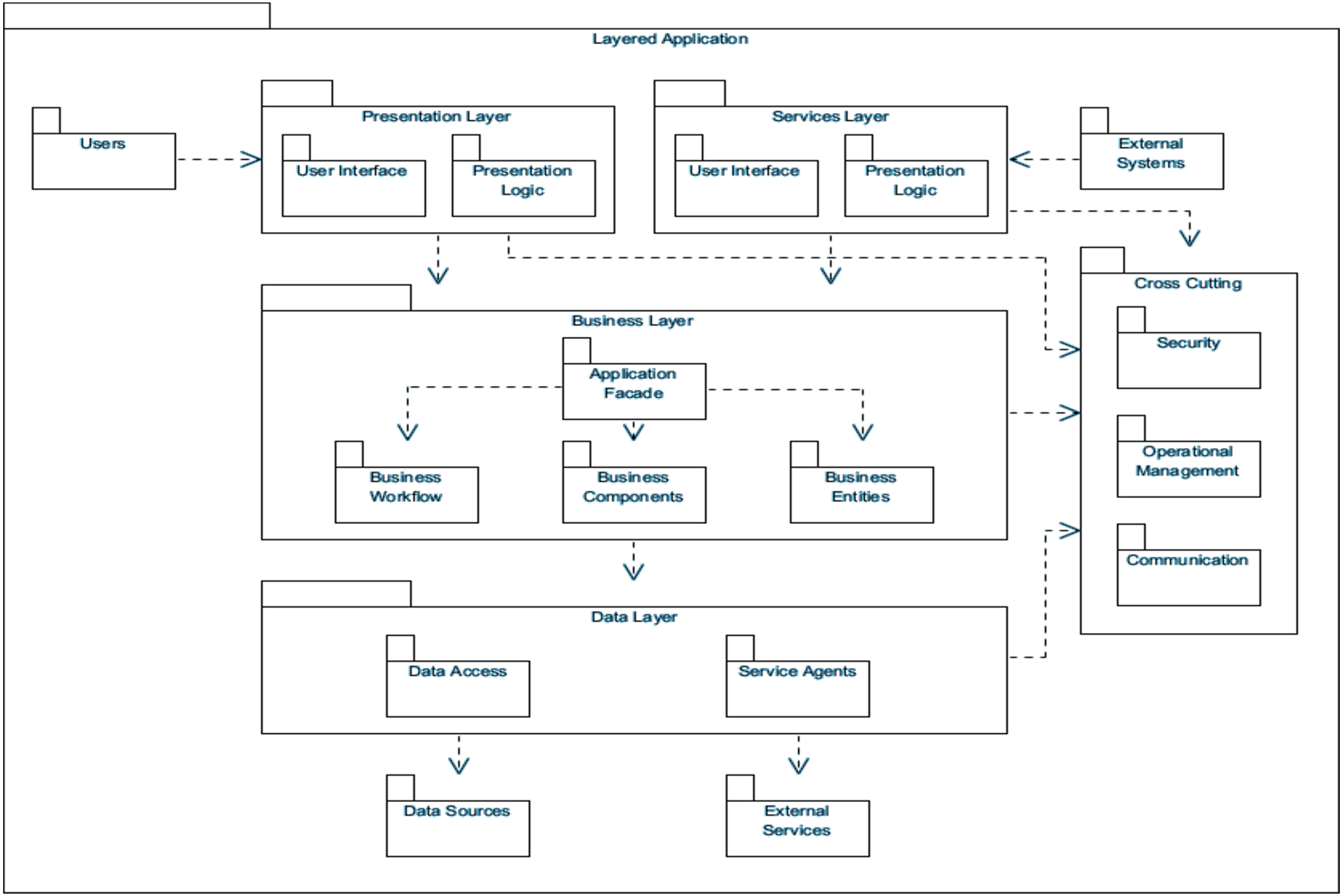
UML: Class Diagram



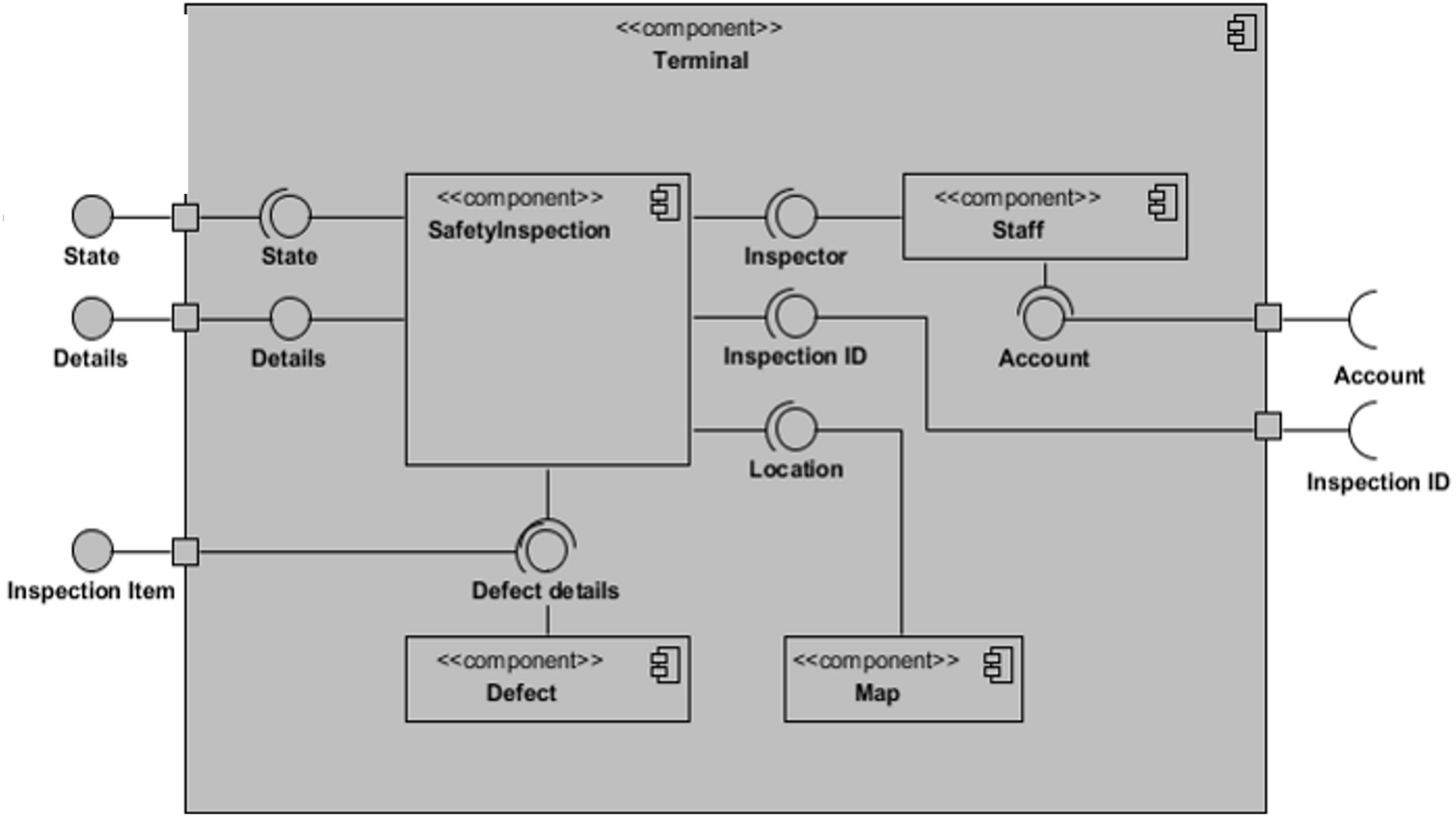

Store
address : Address
name : Text
addSale()
Uses
1
1
ProductCatalog
getSpecification()
Contains
1
*
ProductSpecification
description : Text
price : Money
itemID: itemID
1
Looks-in
1
Houses
1
1
1
Describes
*
Register
endSale()
enterItem()
makePayment()
Captures
1
1
Sale
Date : Date
isComplete : Boolean
time : Time
becomeComplete()
makeLineItem()
makePayment()
getTotal()
Contains
1
*
SaleLineItem
quantity : Integer
getSubtotal()
Payment
amount : Money
*
1
Logs-completed
Paid-by
1

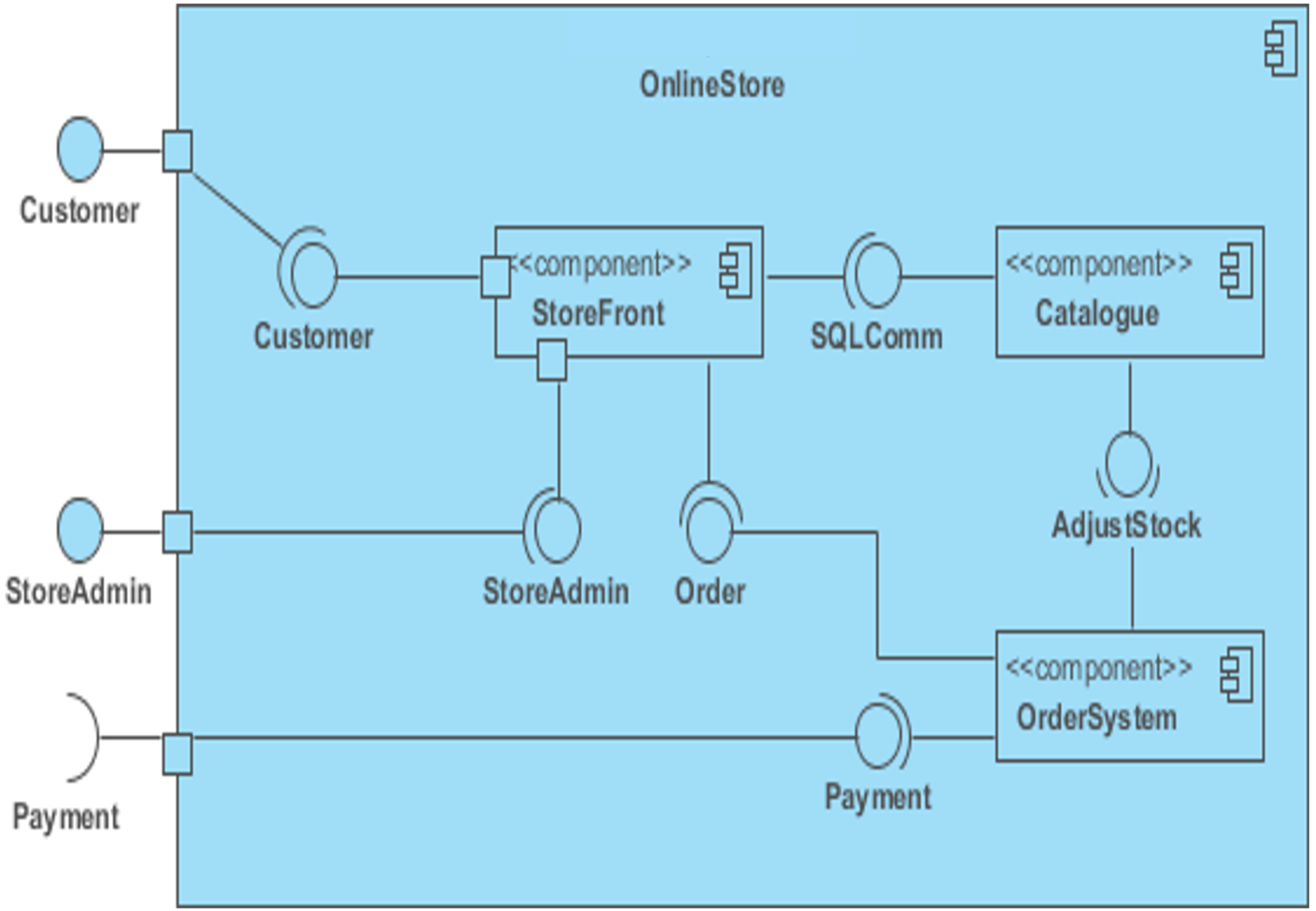
- UML Package Diagram




Layered Application
Users
Presentation Layer
User Interface
Presentation Logic
Services Layer
User Interface
Presentation Logic
External Systems
Cross Cutting
Security
Operational Management
Communication
Business Layer
Application Facade
Business Workflow
Business Components
Business Entities
Data Layer
Data Access
Service Agents
Data Sources
External Services

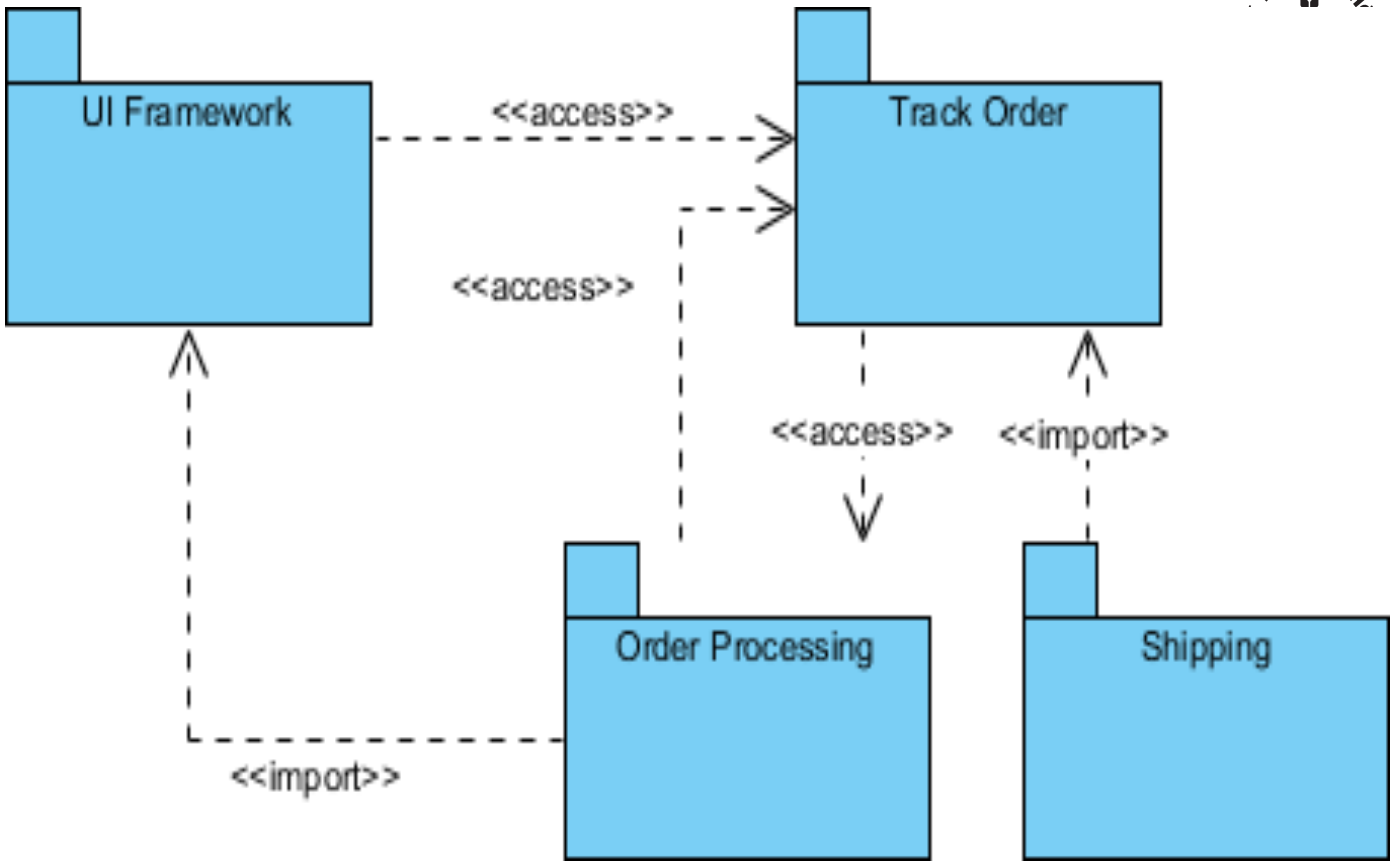
UML Component Diagram



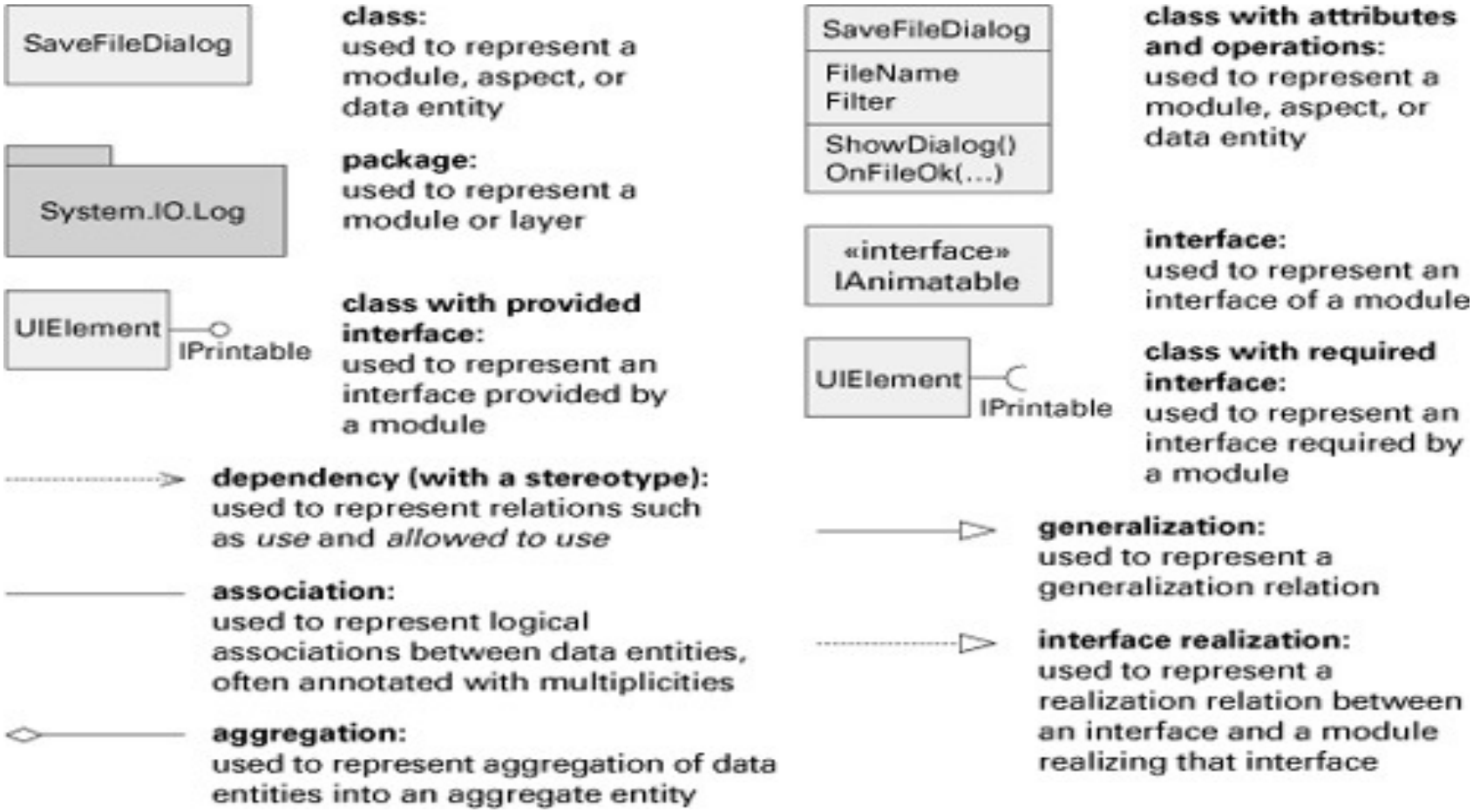

OnlineStore
Customer
Customer
<<component>>
StoreFront
SQLComm
<<component>>
Catalogue
AdjustStock
StoreAdmin
StoreAdmin
Order
<<component>>
OrderSystem
Payment
Payment

Lets check?

- How many module?
- Which one is dependent on which one?
- Does it informs you about the layering of the modules?

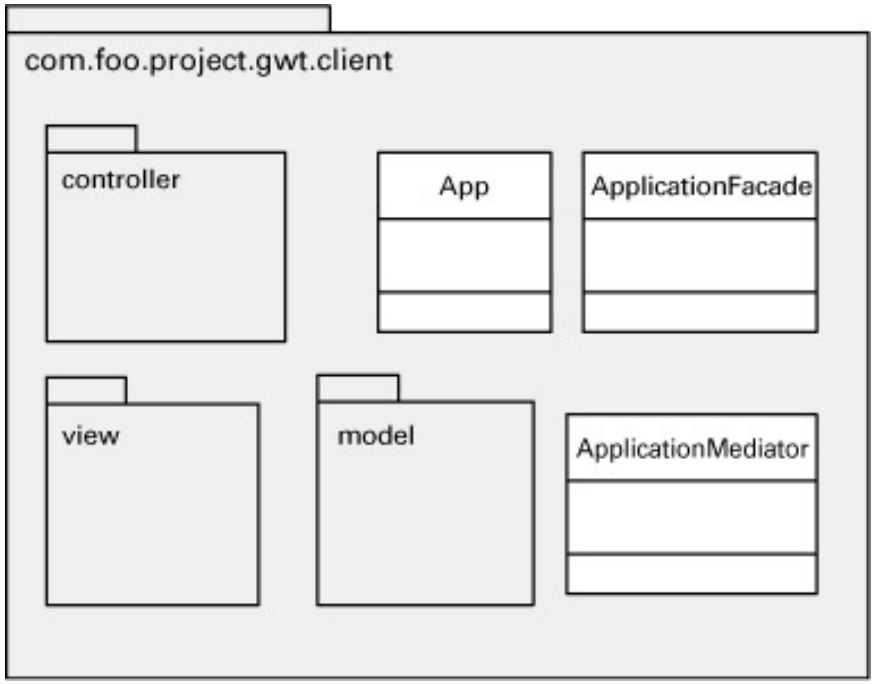


Quick UML symbols primer

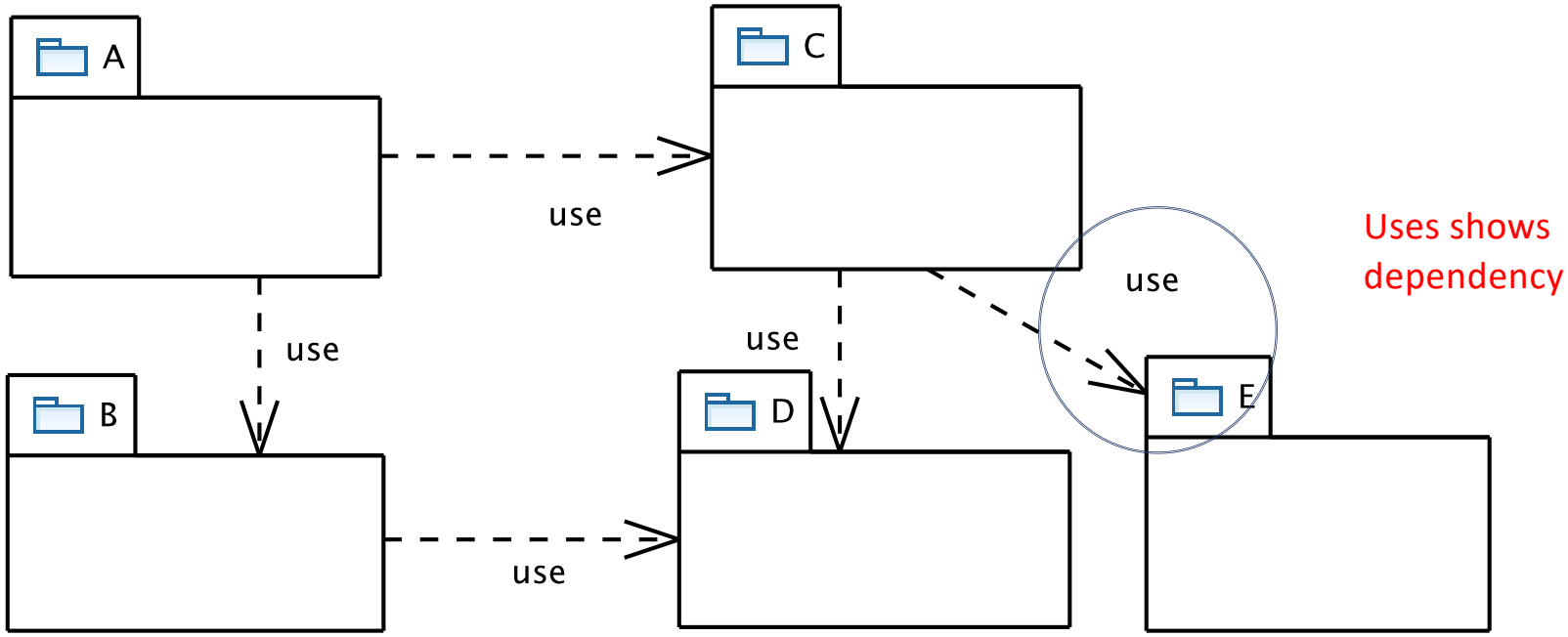


Documenting Module views in UML

- UML Package notion is used to document the module view.
  - or class diagram in some cases
- Modules are normally represented as packages or classes



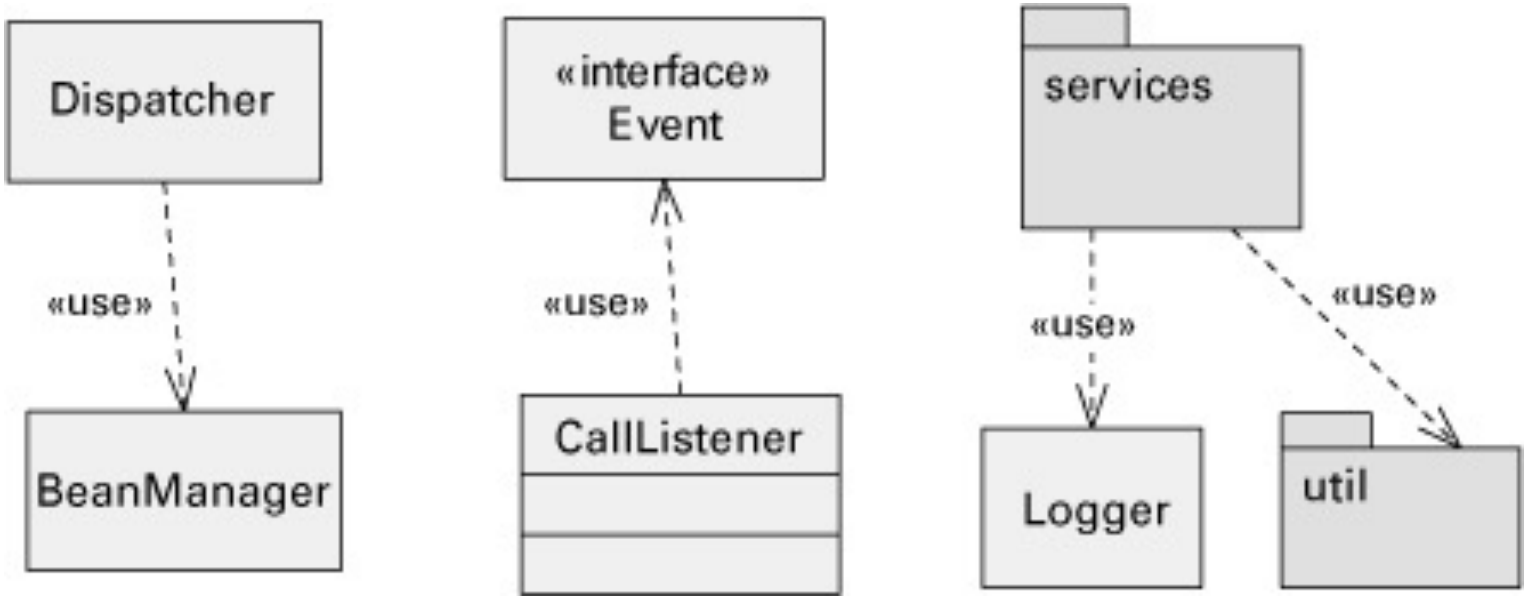
- A module uses B module if it requires part of its exposed functionality
  - A´s correctness **depends_on** B´s correctness



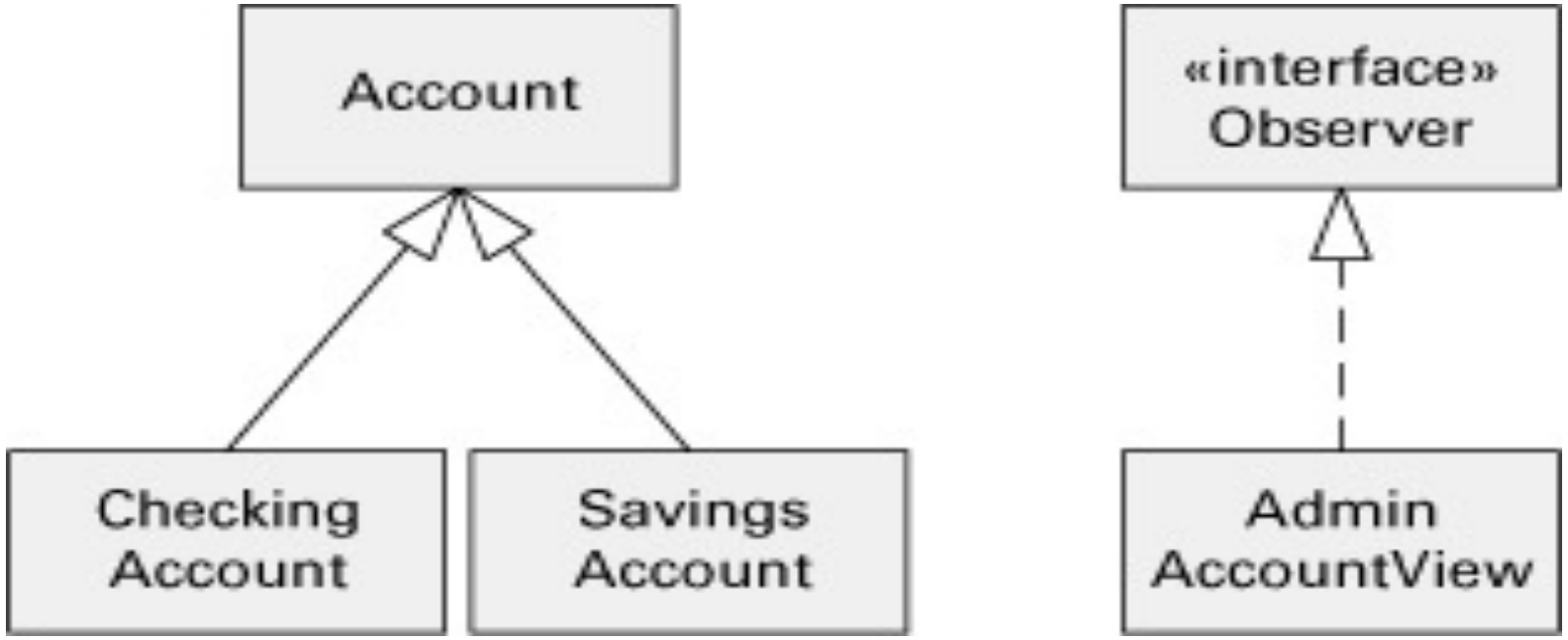
Primary presentation: Module view

- Class, Packages, Interfaces can be represented using UML
- Styles includes:
  - Uses
  - Generalization
  - Layered
  - Aspects
  - Data model

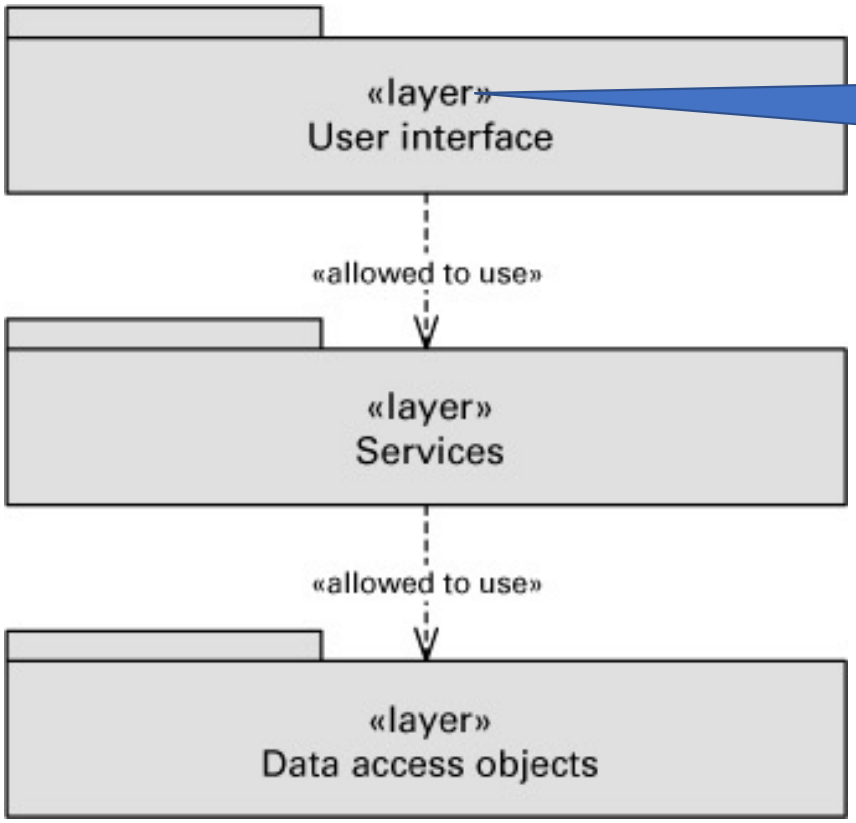
UML: Uses view Style



UML: Class/Generalization view Style



UML: Layered view Style



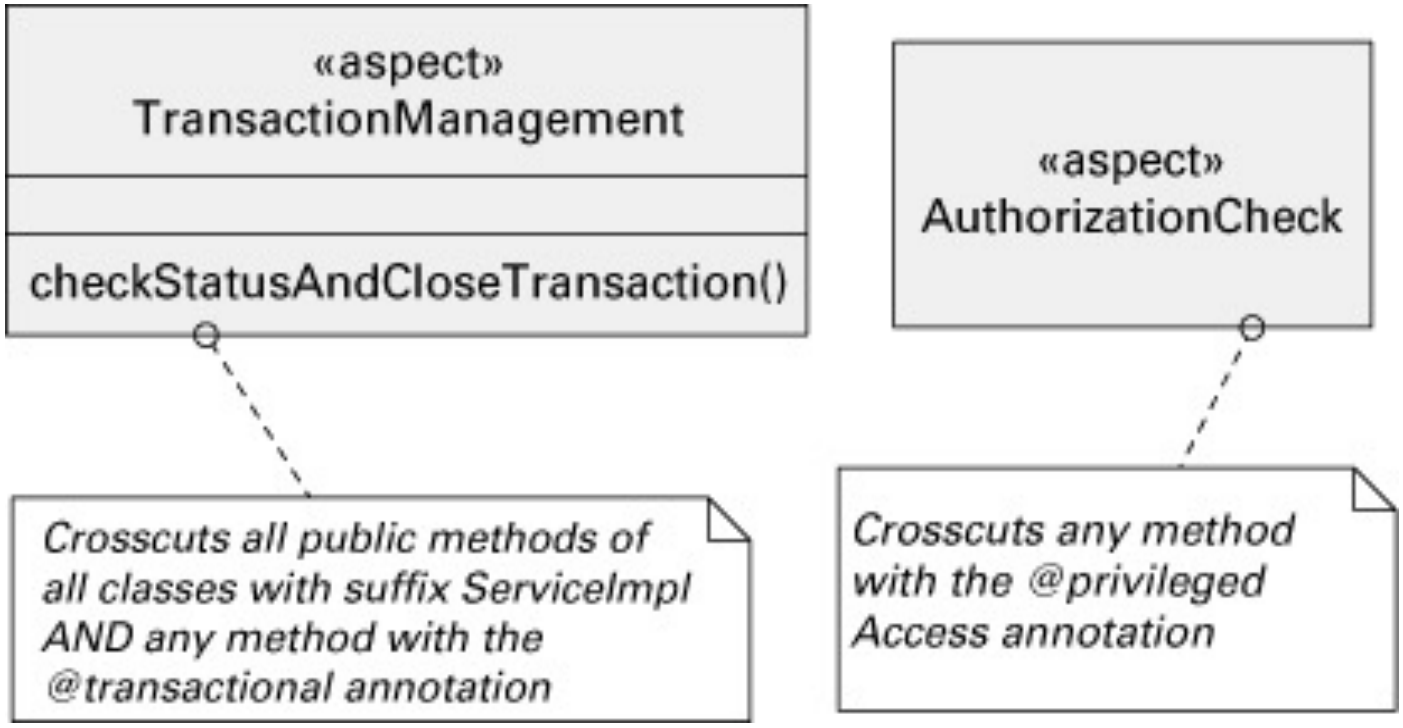
Stereotypes is extensibility mechanisms in UML

You can read more about stereotypes here:

https://www.visual-paradigm.com/tutorials/how-to-create-stereotyped-model-element.jsp

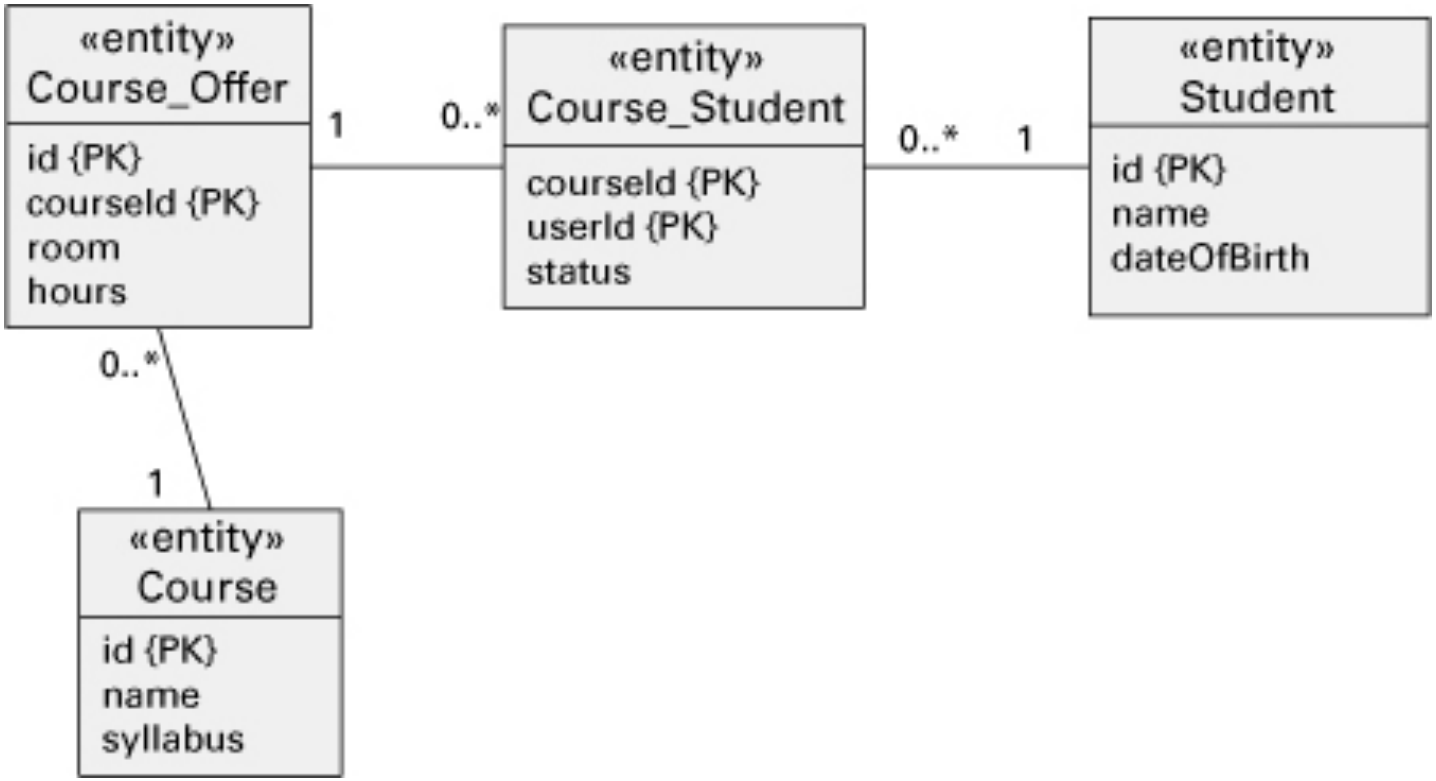
UML: Aspects Style

- Aspect : a module that is responsible for a crosscutting concern (such as internationalization)



UML: Data Model Style

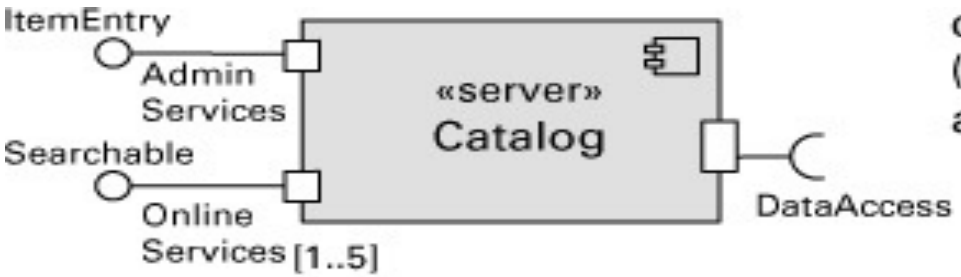
- Class diagram : Classes should have the standard <<entity>> stereotype.



Documenting C&C views in UML

- We use UML to document the view.
- UML Component diagram notion is prime object here.
  - Components: **Runtime elements as Processes and / or Threads**
  - Connectors: Are defined as the **forms of interaction between components** (synchronous, asynchronous, complex transactions)

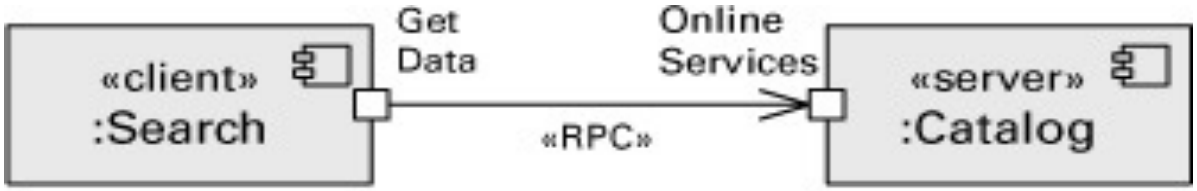
Primary presentation: C&C Views



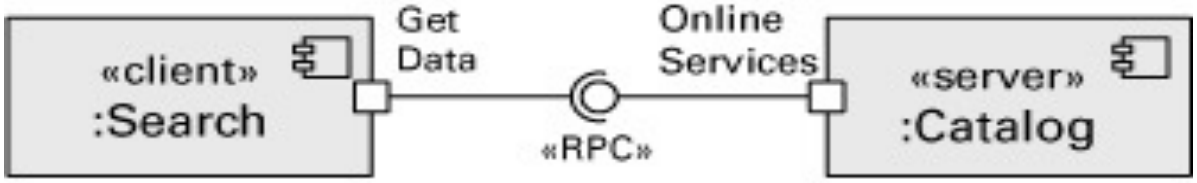
component type (with UML ports and interfaces)

a component instance (with ports explicitly documented)

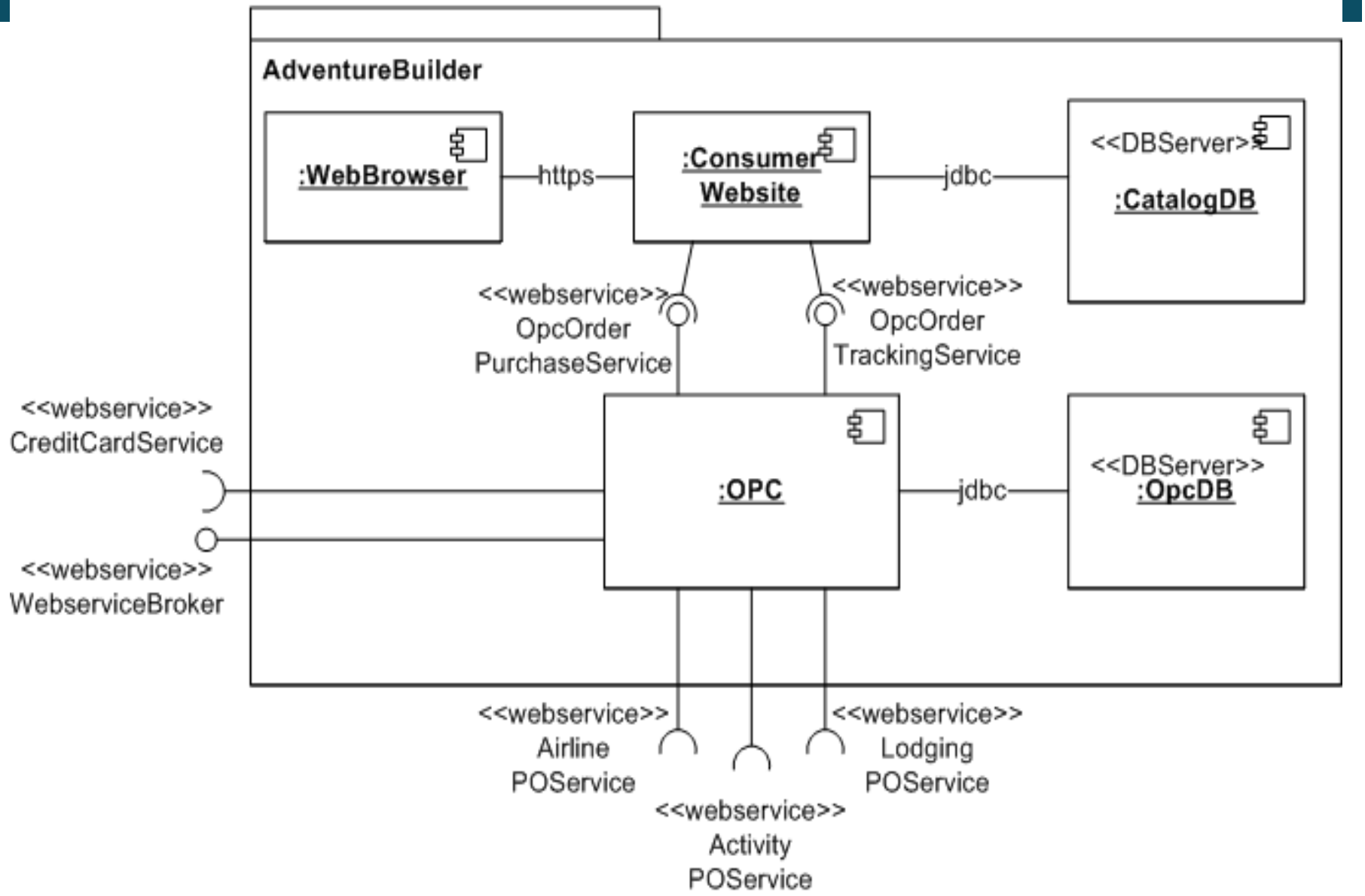
a component instance



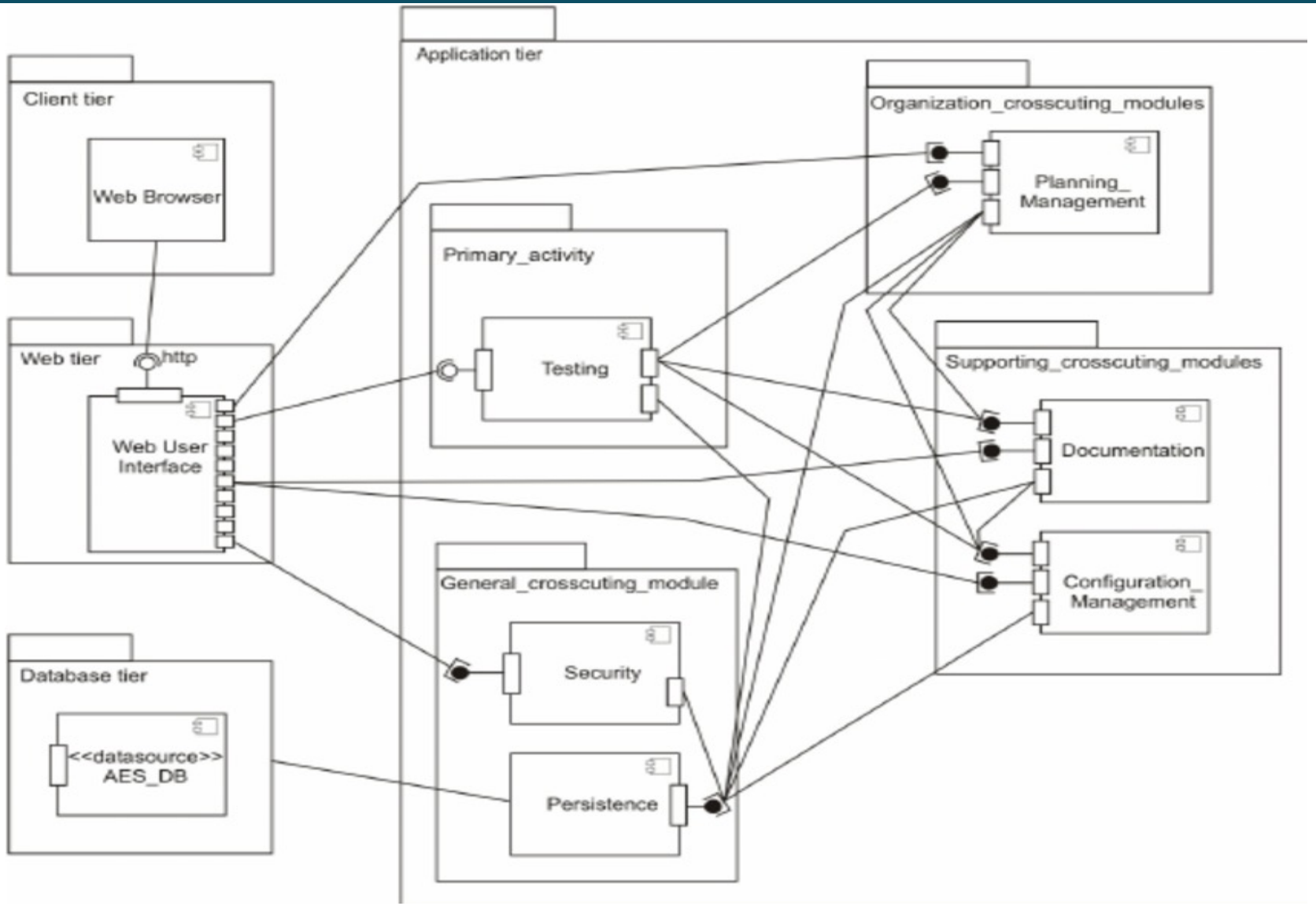
adding navigable end to a connector



using the ball-and-socket notation
for an assembly connector


AdventureBuilder
:WebBrowser
https
:Consumer Website
jdbc
<<DBServer>>
:CatalogDB
<<webservice>>
OpcOrder
PurchaseService
<<webservice>>
OpcOrder
TrackingService
<<webservice>>
CreditCardService
<<webservice>>
WebserviceBroker
:OPC
jdbc
<<DBServer>>
:OpcDB
<<webservice>>
Airline
POService
<<webservice>>
Activity
POService
<<webservice>>
Lodging
POService

- Runtime View of RefTEST

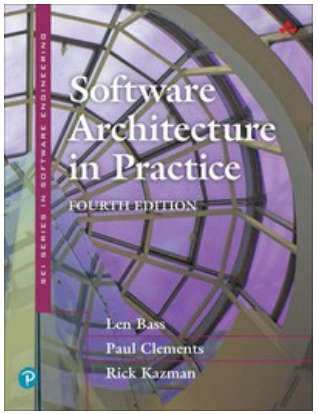


Look up!!

- Package Diagram
  - https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/
- Component Diagram
  - https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
- Composite-structure-diagram
  - https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-composite-structure-diagram/

Reading (Must)

- Chapter 22 of T1: Software Architecture in Practice, 4th Edition, 2021.



https://learning.oreilly.com/library/view/software-architecture-in/9780136885979/

- Chapter 1-4, Reference book: Documenting Software Architectures: Views and Beyond, 2nd Edition, Authors: P. Clements et al., Publisher: Pearson Education, 2010.

Acknowledgment

- Material (diagrams, text etc.) in this lecture is borrowed from the following sources:
  - Bass, Len, Paul Clements, and Rick Kazman. Software architecture in practice. Addison-Wesley Professional, 2021.
  - Documenting Software Architectures: Views and Beyond, 2nd Edition, Authors: P. Clements et al., Publisher: Pearson Education, 2010.
- Learning material and course ideation is provided by Dr. Javier Gonzalez Huerta, Dr. Muhammad Usman & Mr. Ehsan Zabardast


BLEKINGE TEKNISKA HÖGSKOLA
BTH