Dasar Pengembangan Sistem Informasi

Software Design

Tim Pengampu Mata Kuliah
Dasar Pengembangan Sistem Informasi

Topics covered

- Software Design
- Design Quality
- Structured Design
- Object-Oriented Design

Software Engineering Process Activities

Design and implementation

- Software design and implementation is the stage in the software engineering process at which an executable software system is developed.
- Software design and implementation activities are invariably interleaved.
  - Software design is a creative activity in which you identify software components and their relationships, based on a customer's requirements.
  - Implementation is the process of realizing the design into a program.

Build or Buy?

- In a wide range of domains, it is now possible to buy off-the-shelf systems (COTS) that can be adapted and tailored to the users' requirements.
- When you develop an application in this way, the design process becomes concerned with how to use the configuration features of that system to deliver the system requirements.

For example, if you want to implement a medical records system, you can buy a package that is already used in a hospital. It is cheaper and faster to deploy.

Purpose of Design

- Design is where customer requirements, business needs, and technical considerations all come together in the formulation of a product or system
- The design model provides detail about the software data structures, architecture, interfaces, and components

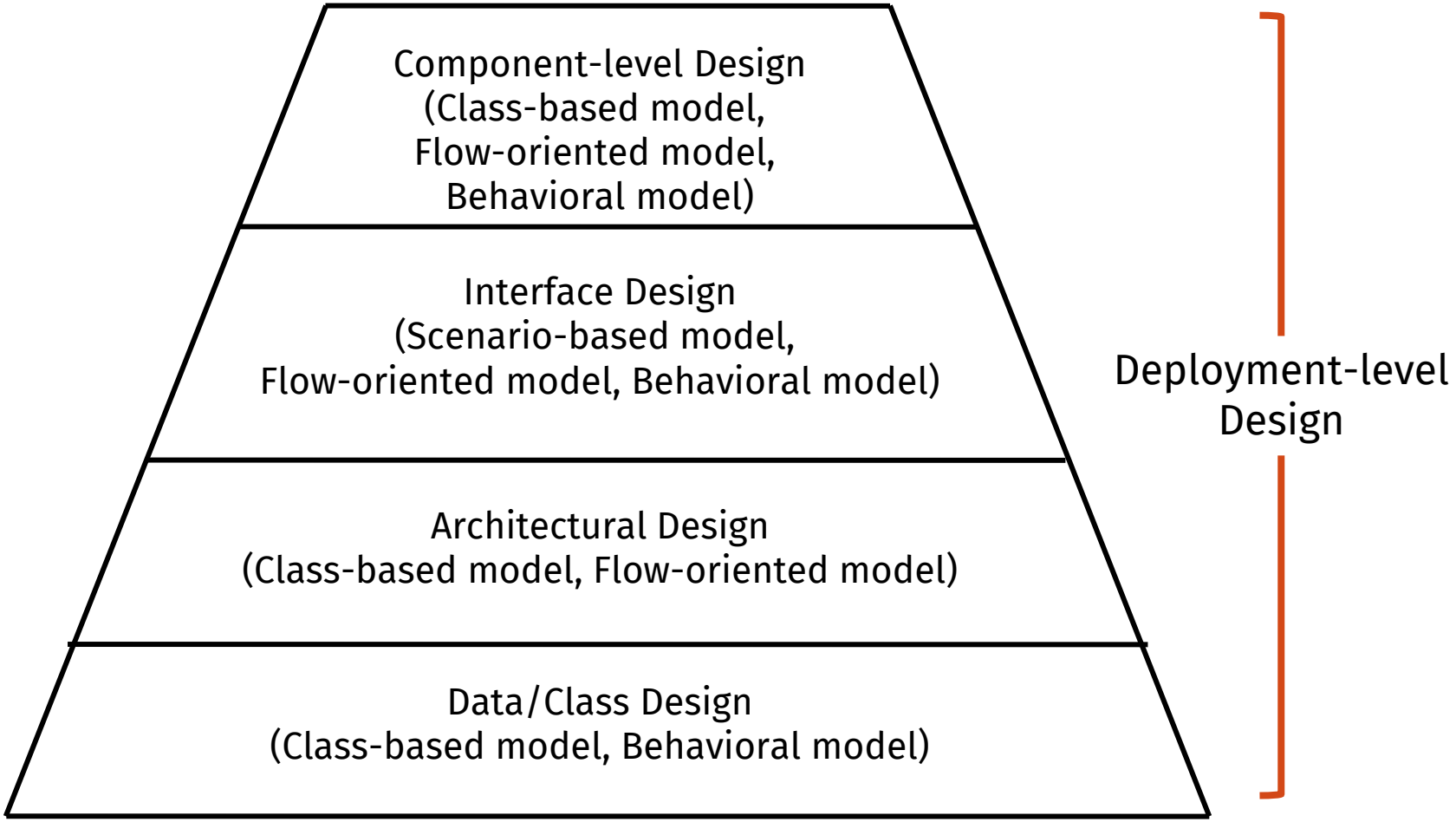
How to Design?

- The design model can be assessed for quality and be improved before code is generated and tests are conducted
- A designer must practice diversification and convergence
  - Find as much alternatives as possible, then choose the suitable one
- Software design is an iterative process through which requirements are translated into a blueprint for constructing the software
  - Begin from high-level design to detailed-level design

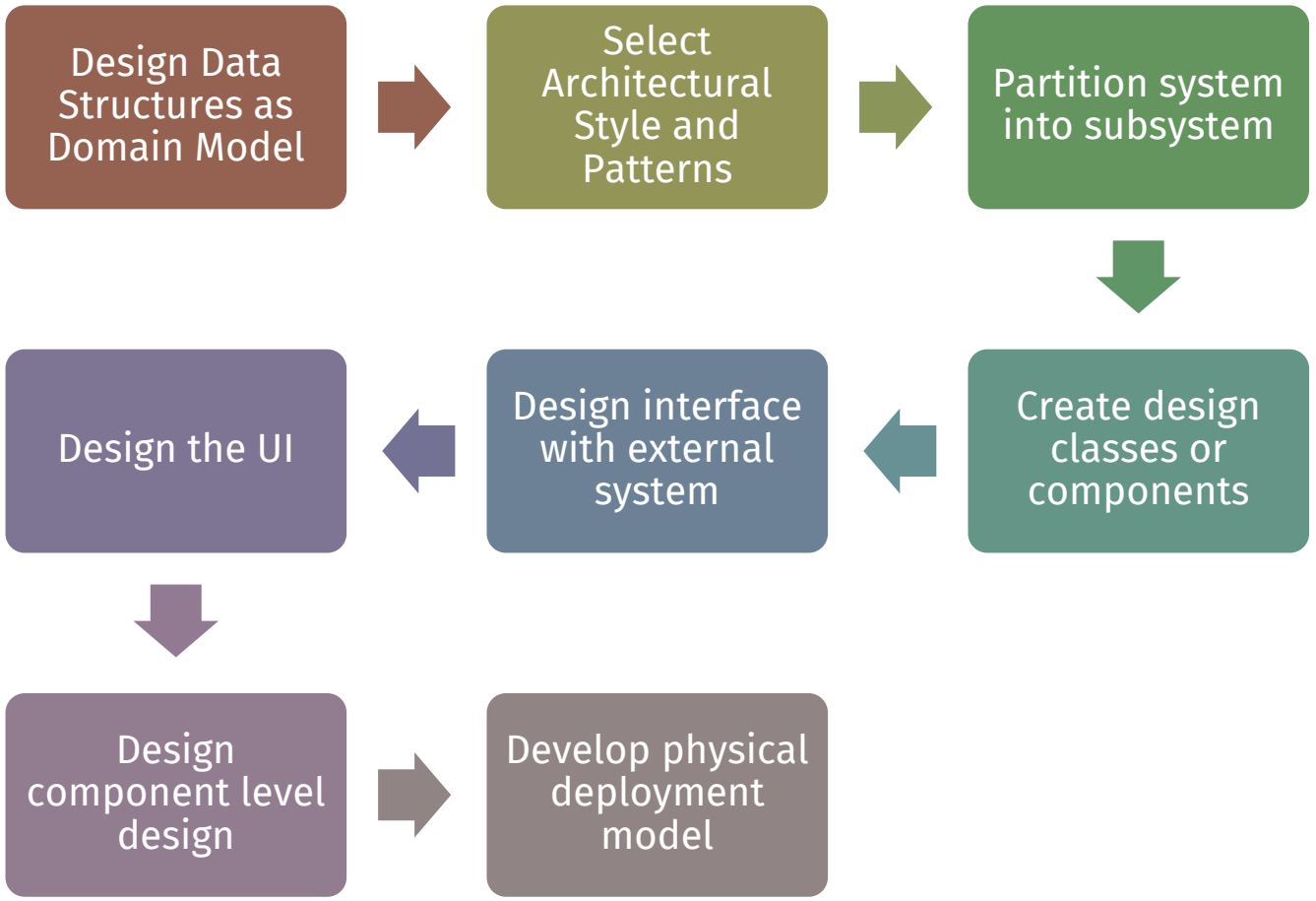
From Analysis Model to Design Model

- The data/class design transforms analysis classes into design classes along with the data structures required to implement the software
- The architectural design defines the relationship between major structural elements of the software; architectural styles and design patterns help achieve the requirements defined for the system
- The interface design describes how the software communicates with systems that interoperate with it and with humans that use it
- The component-level design transforms structural elements of the software architecture into a procedural description of software components

From Analysis Model to Design Model



Task Set for Software Design



Design Quality

Quality's Role

- The importance of design is quality
- Design is the place where quality is fostered
  - Provides representations of software that can be assessed for quality
  - Accurately translates a customer's requirements into a finished software product or system
  - Serves as the foundation for all software engineering activities that follow

Quality's Role

- Without design, we risk building an unstable system that:
  - Will fail when small changes are made
  - May be difficult to test
  - Cannot be assessed for quality later in the software process when time is short and most of the budget has been spent
- The quality of the design is assessed through a series of formal technical reviews or design walkthroughs

Goals of a Good Design

- The design must implement all of the explicit requirements contained in the analysis model
  - It must also accommodate all of the implicit requirements desired by the customer
- The design must be a readable and understandable guide for those who generate code, and for those who test and support the software
- The design should provide a complete picture of the software, addressing the data, functional, and behavioural domains from an implementation perspective

FILKOM
FAKULTAS ILMU KOMPUTER

Design Methodologies

Well-known Software Development Methodologies

- Structured
  - Based on process approach
- Object-Oriented
  - Based on object/data approach

Structured

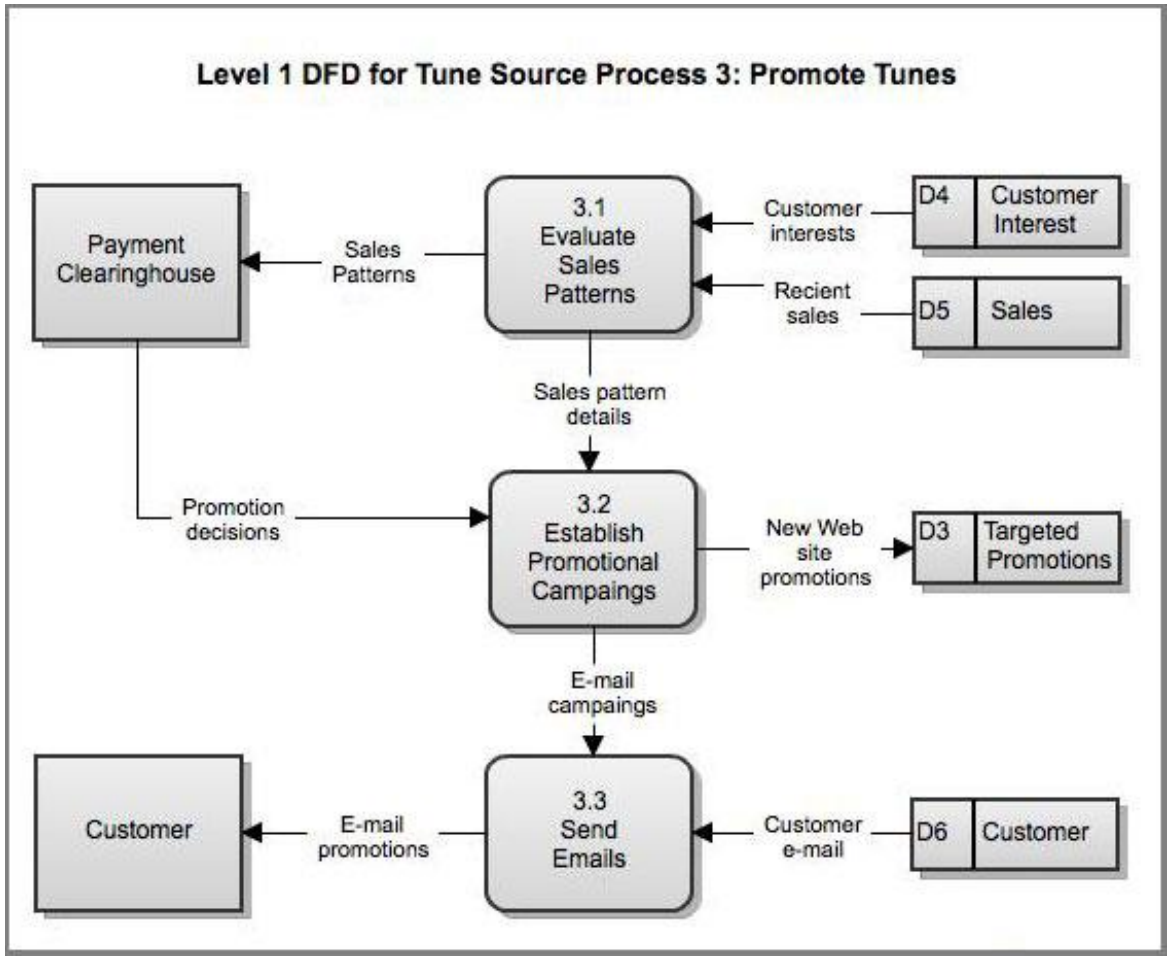
Structured Methodologies

- Applies standard SDLC,
  - gives a measure of control, but little help in improving the productivity and quality of analysis and design.
- 1970's: structured methodologies developed
  - promotes more effective analysis and more stable /maintainable designs.
  - more largely process-oriented:
  - minor on modeling of entities and data

Tools for Structured Methodologies

- Dataflow diagram (DFD): Process/flow of data
- Data Dictionary: definition, ”encyclopedias”
- Entity relationship diagram (ERD):
- Hierarchy diagram: simple top-down connection
- Software-spec: English specification of logic, process
- State-transition diagram: possible states
- Structure chart: architecture of system

Example: DFD


Level 1 DFD for Tune Source Process 3: Promote Tunes
Payment Clearinghouse
Sales Patterns
3.1 Evaluate Sales Patterns
Customer interests
D4 Customer Interest
Recient sales
D5 Sales
Sales pattern details
Promotion decisions
3.2 Establish Promotional Campaings
New Web site promotions
D3 Targeted Promotions
E-mail campaings
Customer
E-mail promotions
3.3 Send Emails
Customer e-mail
D6 Customer

Example: Data Dictionary

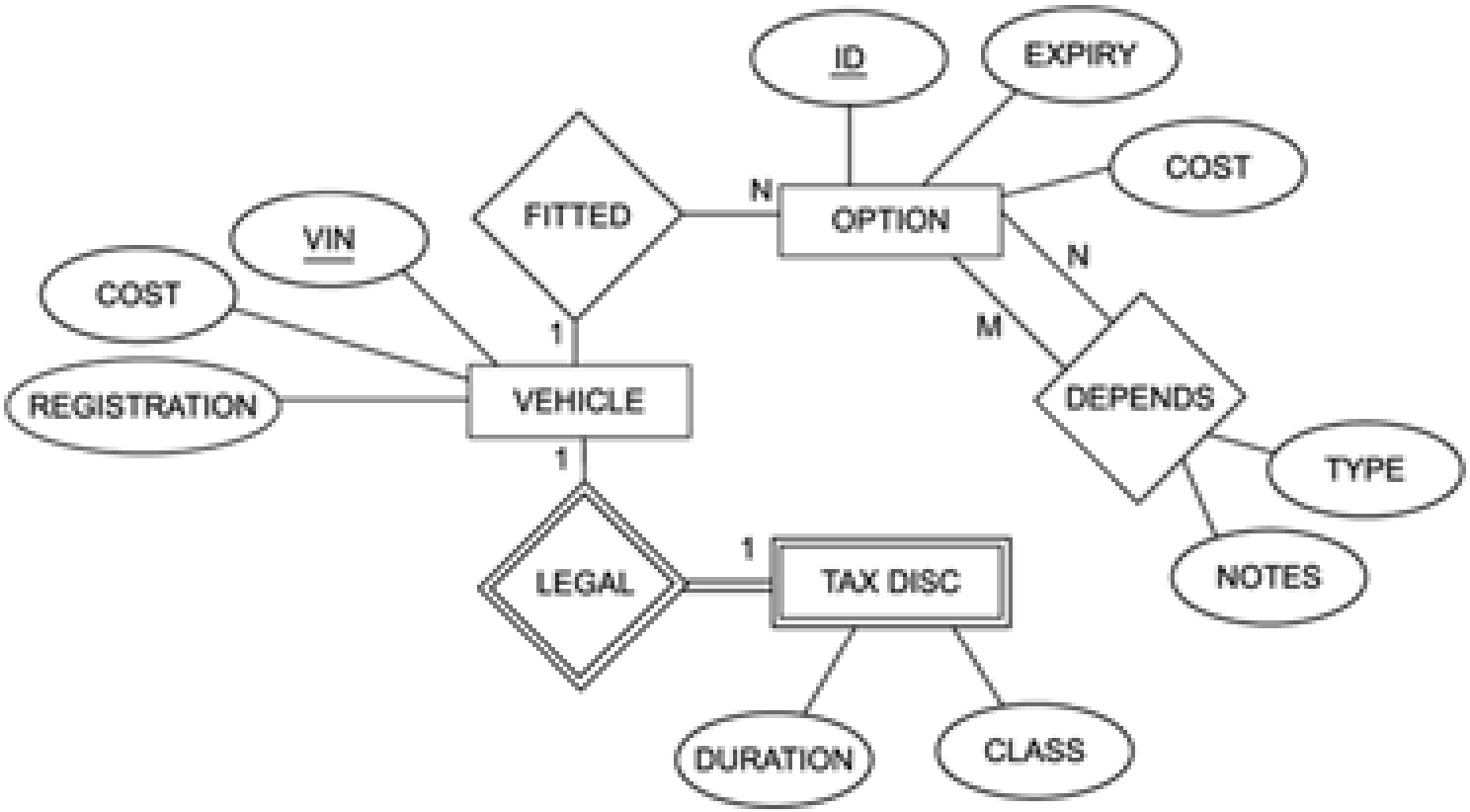
Presentation

Field Name	Data type	Field Length	Constrain	Description
p_id	Number	10	Primary key	Presentation id,Auto generated
Pres_name	Varchar	20	Foreign key	Name of presentation
Whiteboard_no	Number	10		Number of multiple whiteboard
Total_counter	Number	10	Not null	Total no. of slide

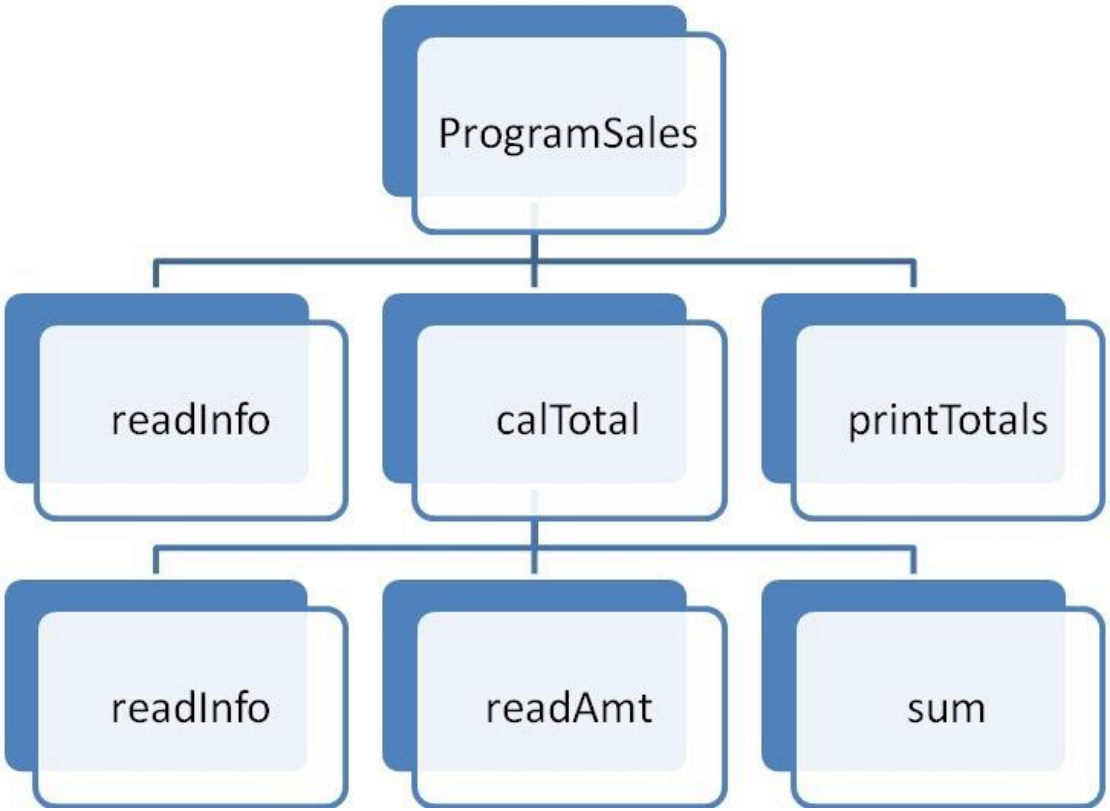
Message

Field Name	Data type	Field Length	Constrain	Description
m_id	Number	10	Primary key	Message id ,Auto generated
user_id	Number	10	Foreign key	Id of user
msg	varchar2	200	Not null	Any message

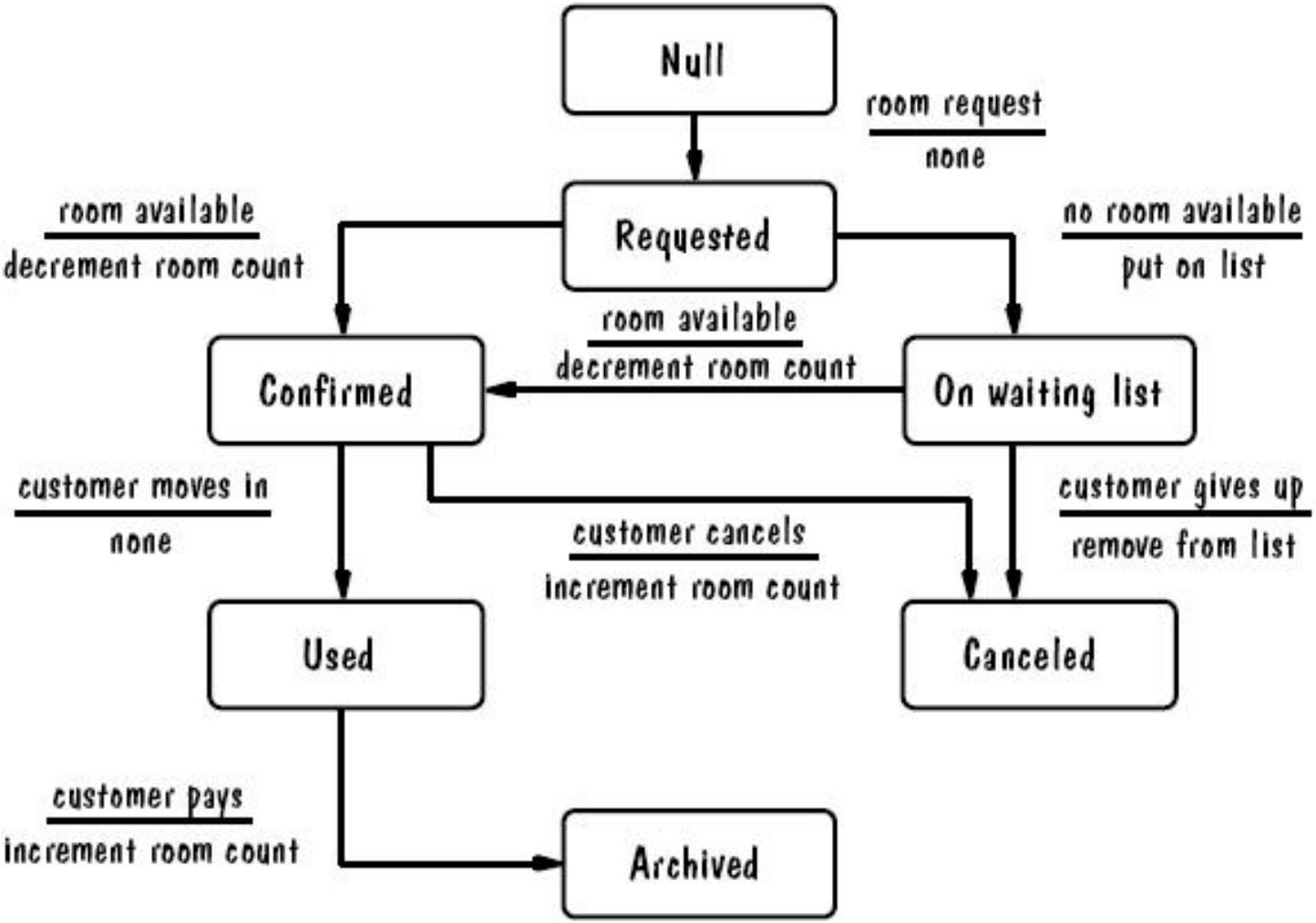
Example: ER Diagram



Example: Hierarchy Diagram



Example: State Transition Diagram



Object-Oriented

Object-Oriented Methodology

- Objects are abstractions of real-world or system entities and manage themselves
- Objects are independent and encapsulate (have) state and representation information.
- System functionality is expressed in terms of object services
- Shared data areas are eliminated, Objects communicate by message passing
- Objects may be distributed and may execute sequentially or in parallel

Advantages of OOD

- Easier maintenance.
  - Objects may be understood as stand-alone entities
- Objects are appropriate reusable components
- For some systems, there may be an obvious mapping from real world entities to system objects

Object-oriented Development

- Object-oriented analysis (OOA), design (OOD) and programming (OOP) are related but distinct
- OOA concerned with developing an object model of the application domain
- OOD concerned with developing an object-oriented system model to implement requirements
- OOP concerned with realising an OOD using an OO programming language such as Java or C++

Object-Oriented Concepts

- Object and Class
- Messages
- Generalization and Inheritance
- Abstract Class
- Interface
- Polymorphism

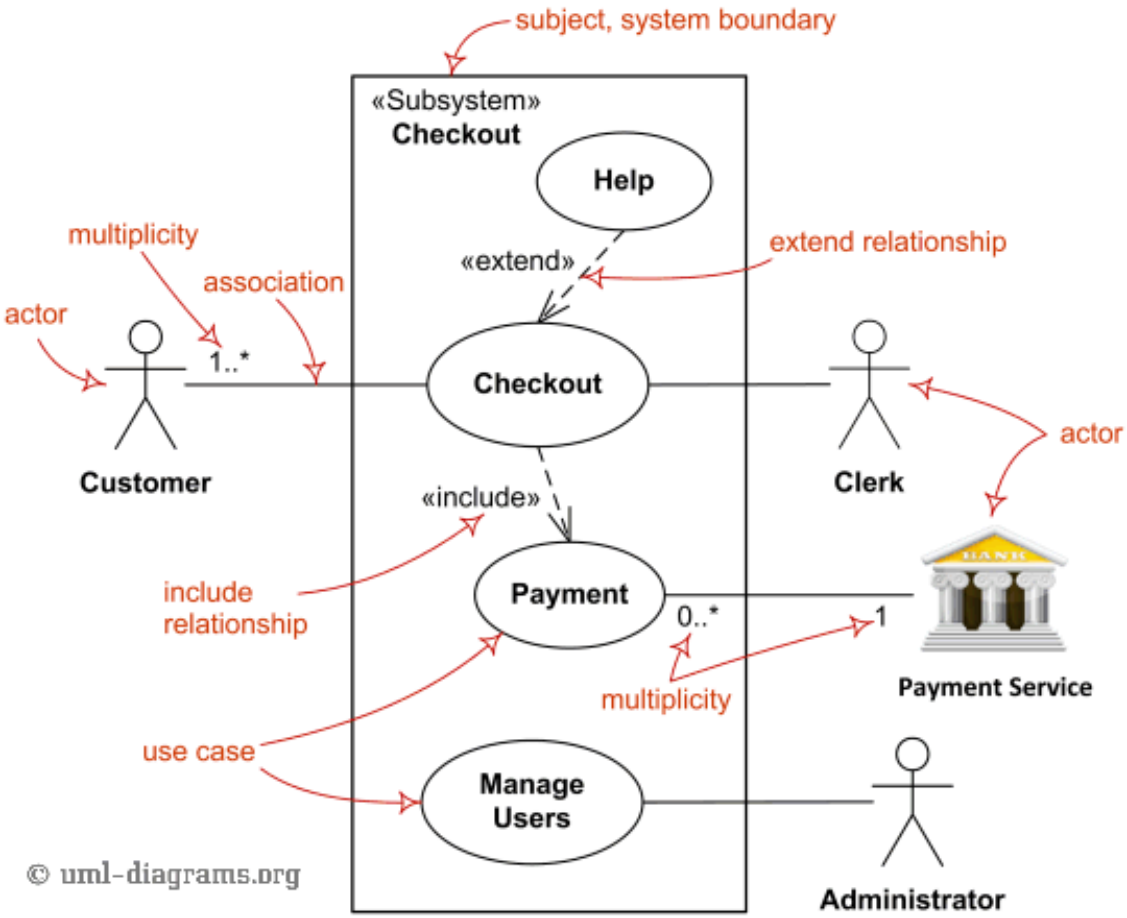
The Unified Modeling Language (UML)

- Several different notations for describing object-oriented designs were proposed in the 1980s and 1990s
- The Unified Modeling Language is an integration of these notations
- It describes notations for a number of different models that may be produced during OO analysis and design
- It is now a de facto standard for OO modelling

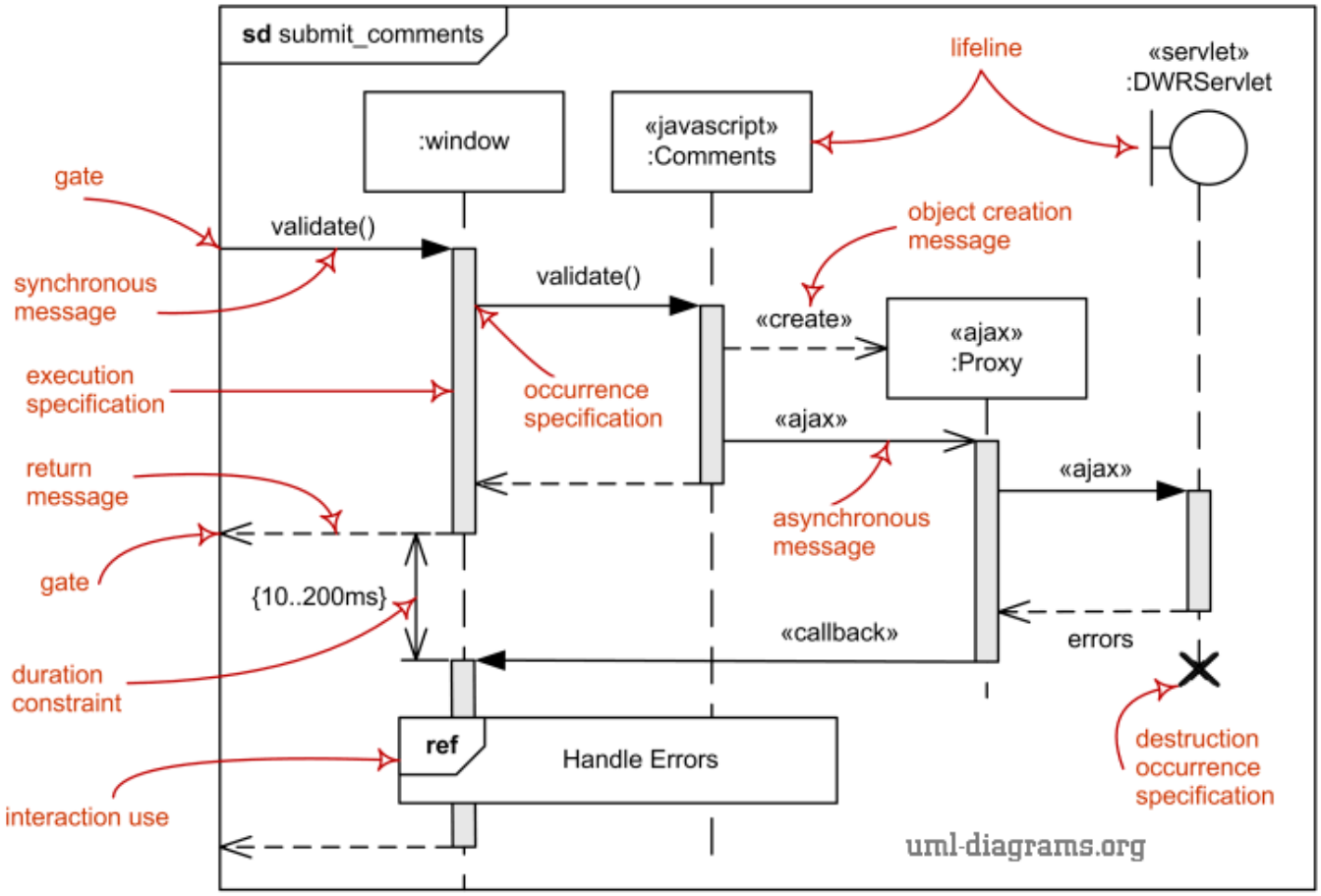
Object-oriented Design Process

1. Define the context and modes of use of the system
   - Use Case Diagram, Use Case Description, and Specification
2. Design the system architecture
   - Component Diagram, Package Diagram
3. Identify the principal system objects
   - Class Diagram
4. Develop design models
   - Class Diagram and Relationships, Sequence Diagram, State Chart Diagram
5. Specify object interfaces
   - Interface diagram, UI/UX and Layout Design

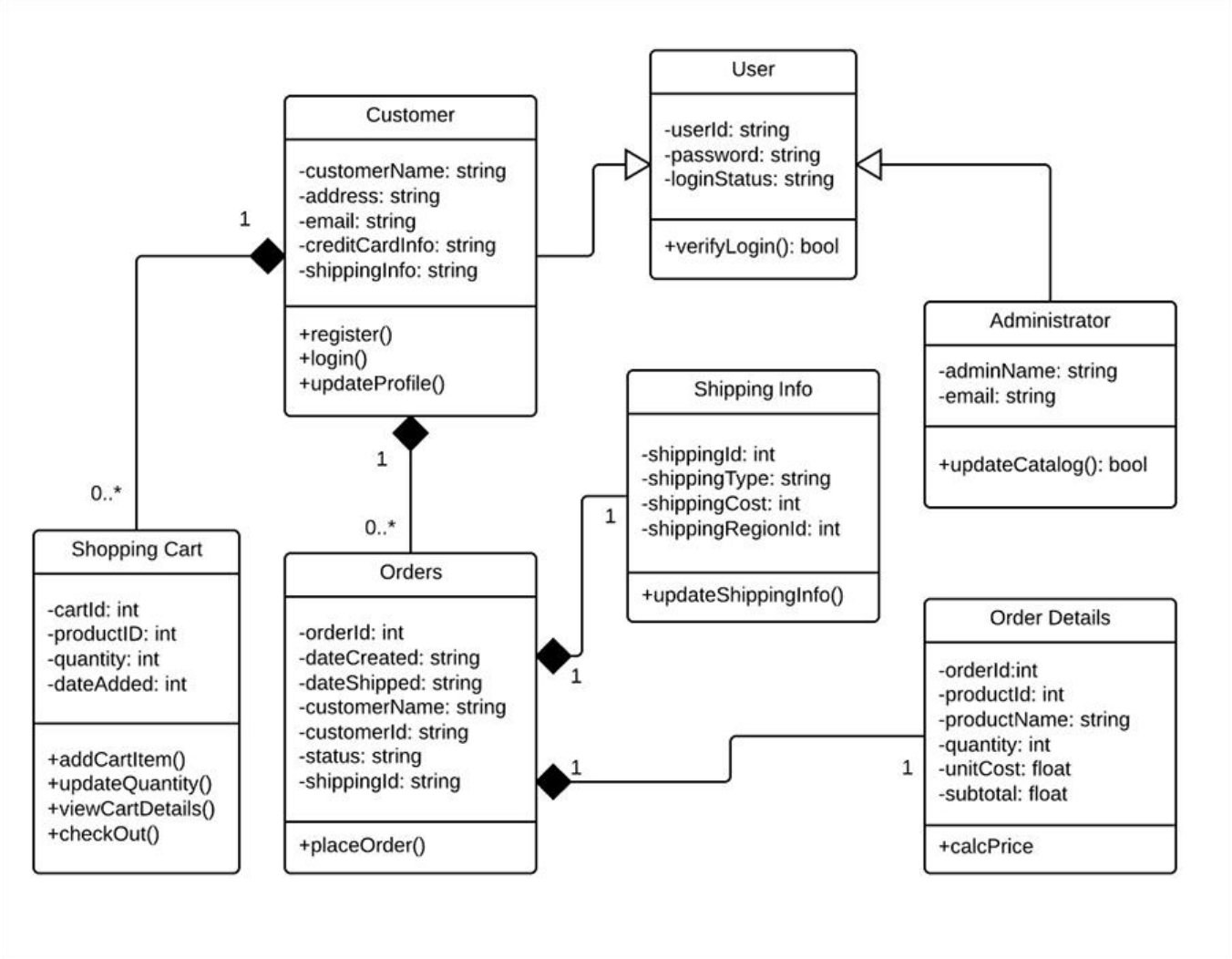
Example: Use Case Diagram



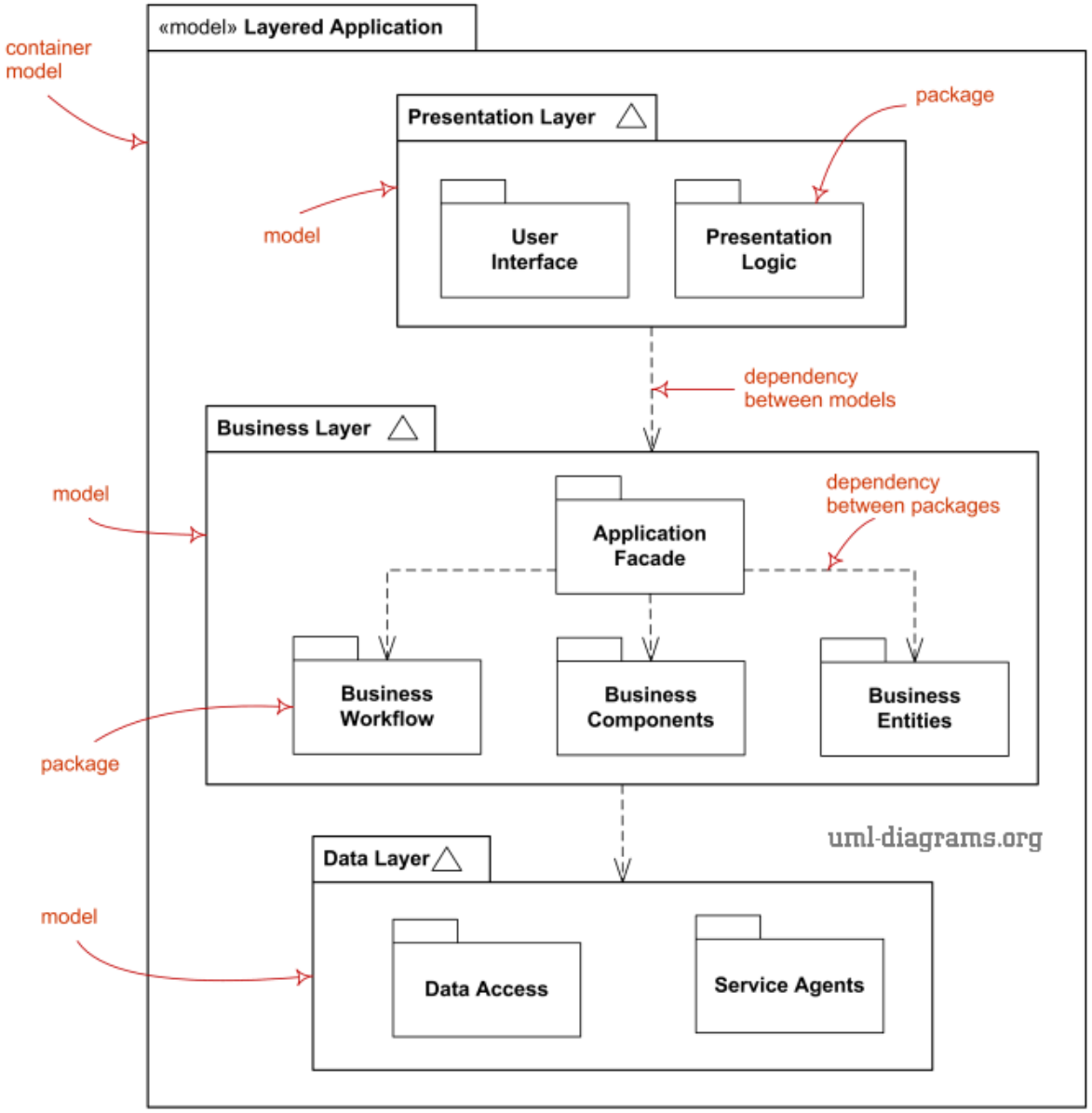
Example: Sequence Diagram



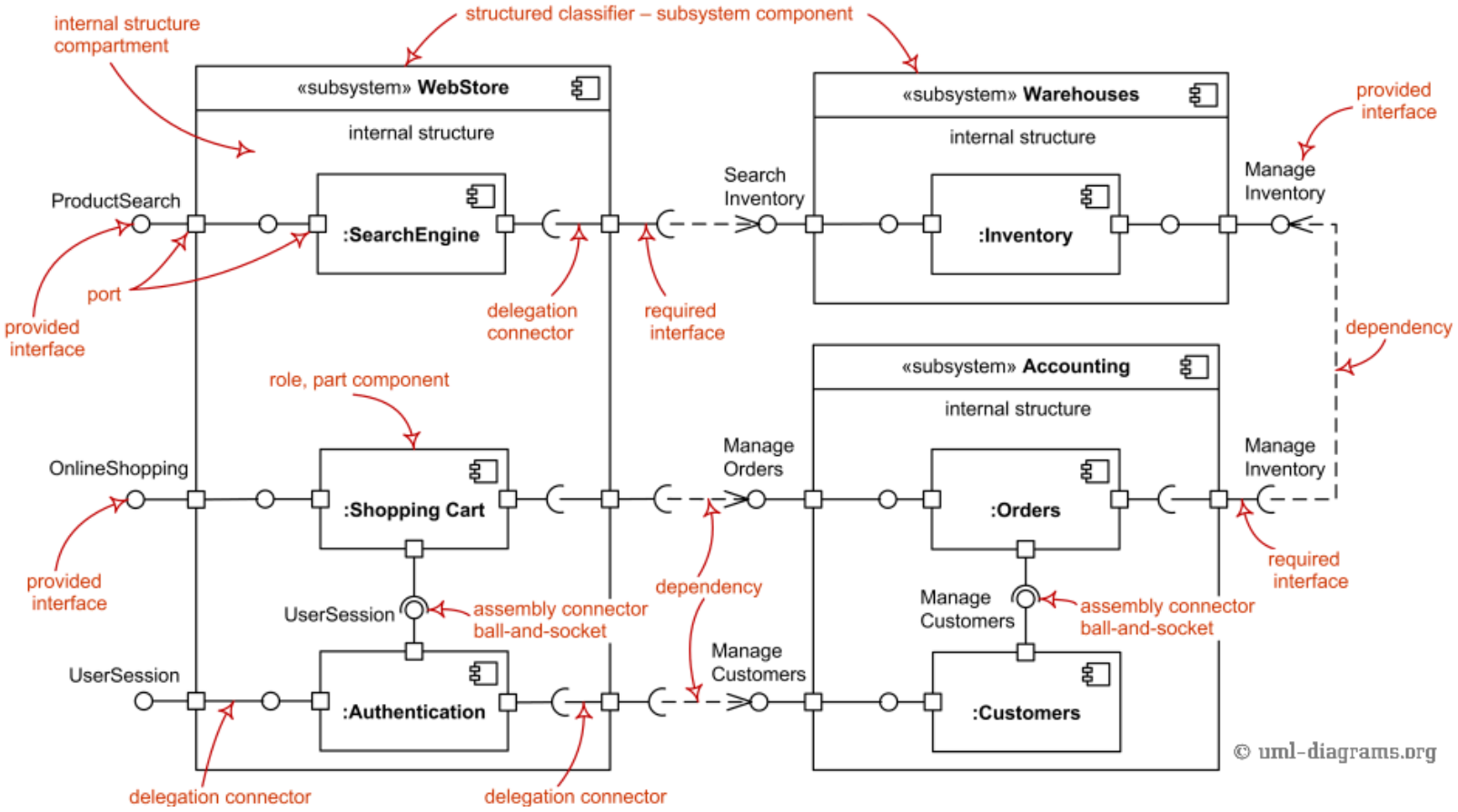
Example: Class Diagram



Example: Package Diagram



Example: Component Diagram



Questions?