DIS - Exercise 7

Artifact 6

2021/2022

Martin Radvanský

Artifact 5 - **Domain model design**

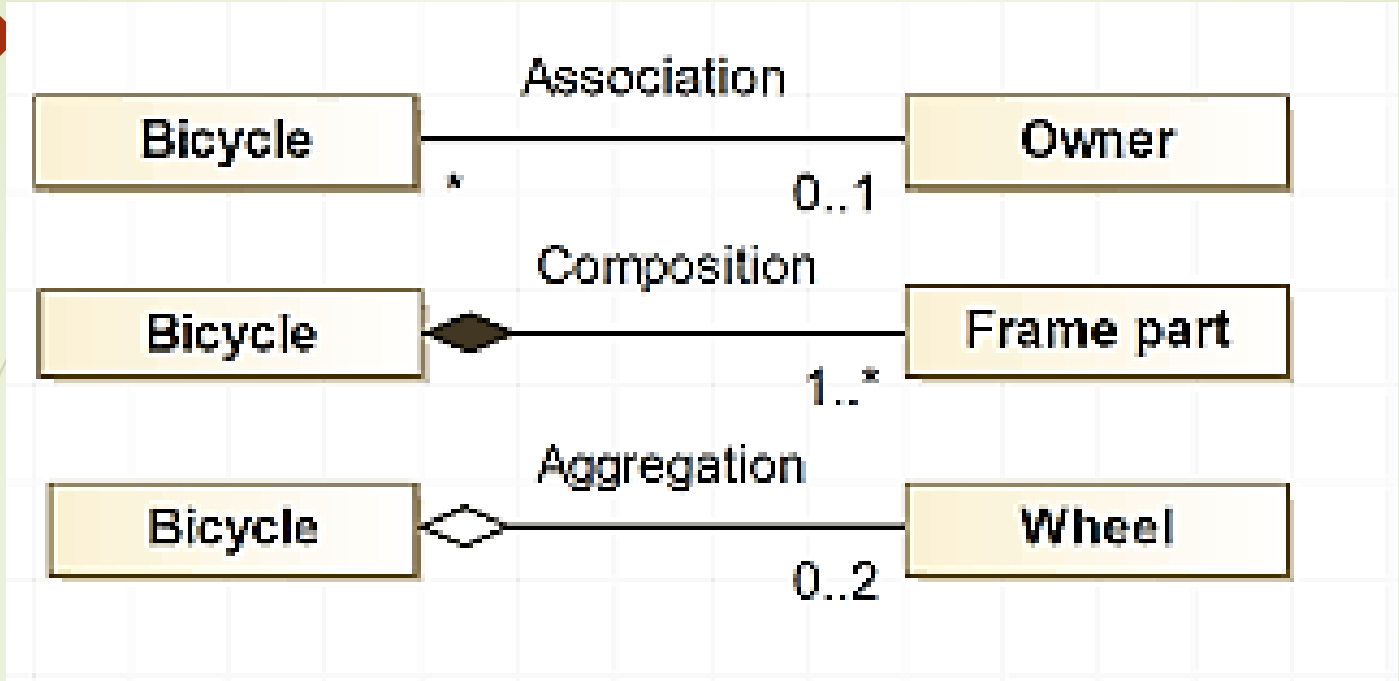
- **Domain model -** extended conceptual model. Classes (methods, interface, data types)
- Relationships (completion)
- Interactions - static class diagram
- Sequence diagrams for the chosen functionality implementation.
- Design patterns (and architectural) used in the data, domain or presentation layer.

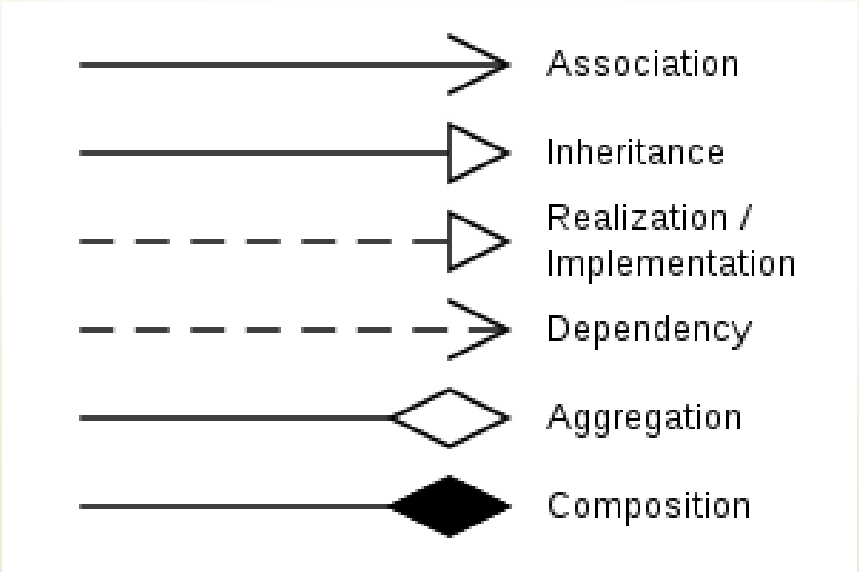
Artifact 5 - **Domain model design**

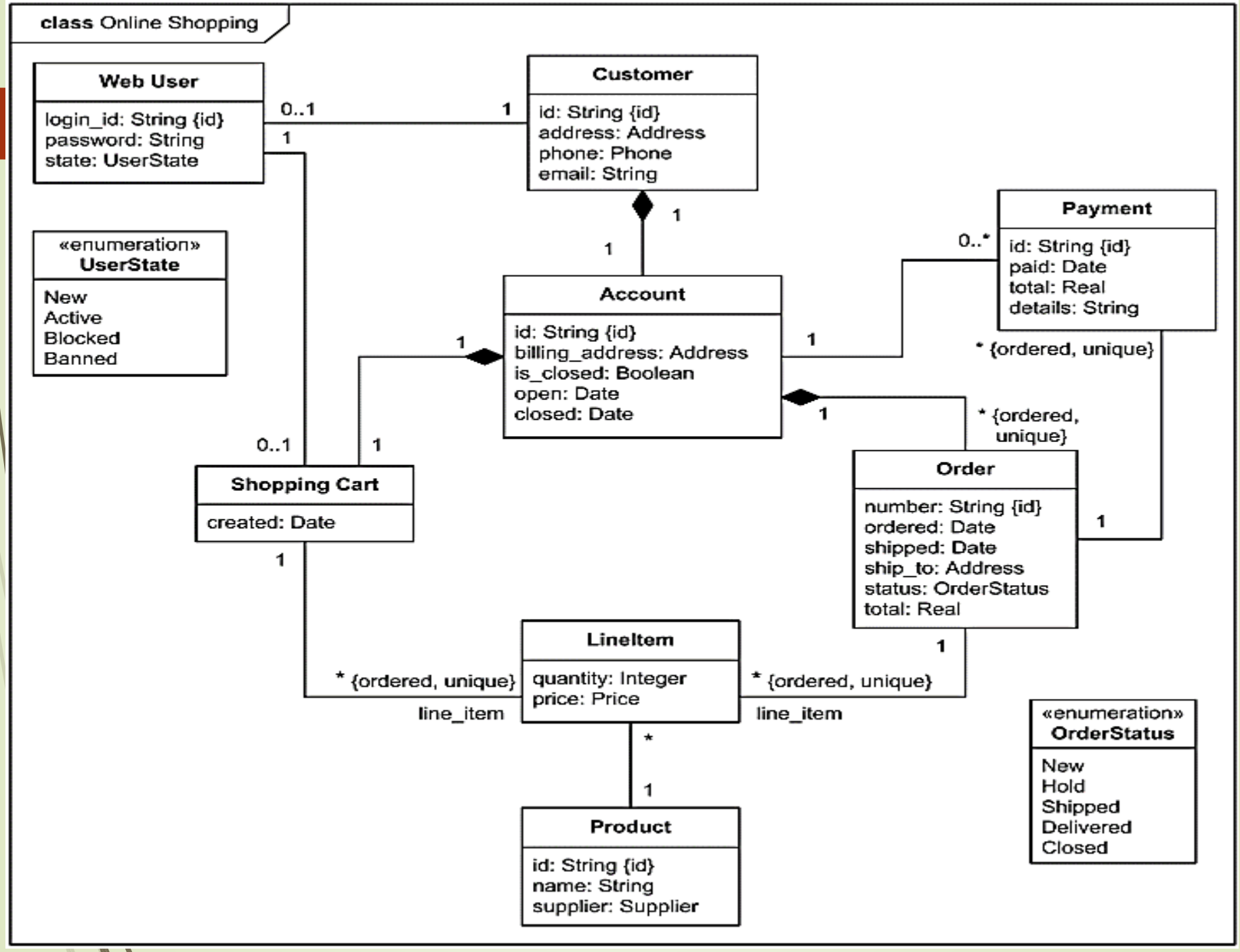
Static class diagram (data + methods), association types, inheritance.

- Patterns used in domain model.
- It is possible to separate in diagrams the pure domain design from the design with patterns. **Separate layers PL, BL, DAL for better readability.**
- Sequence diagrams for key operations (especially object cooperation)

Relationship between classes






class Online Shopping
Web User
login_id: String {id}
password: String
state: UserState
0..1
1
1
Customer
id: String {id}
address: Address
phone: Phone
email: String
1
1
«enumeration»
UserState
New
Active
Blocked
Banned
Payment
id: String {id}
paid: Date
total: Real
details: String
0..*
Account
id: String {id}
billing_address: Address
is_closed: Boolean
open: Date
closed: Date
1
1
1
1
* {ordered, unique}
* {ordered, unique}
0..1
1
Shopping Cart
created: Date
1
Order
number: String {id}
ordered: Date
shipped: Date
ship_to: Address
status: OrderStatus
total: Real
1
1
LineItem
quantity: Integer
price: Price
* {ordered, unique}
line_item
* {ordered, unique}
line_item
*
1
Product
id: String {id}
name: String
supplier: Supplier
«enumeration»
OrderStatus
New
Hold
Shipped
Delivered
Closed

Artifact 5 - Content

- **Static Class Diagram** (divided it into **three diagrams** by the layer – PL-BL-DAL). Use patterns you have learned in lectures.
  - **GoF patterns** - Singleton, Gateway, Proxy, etc.
  - **Enterprise patterns** - Lazy load, Identity field, Foreign key mapping, Embedded value, Inheritance mapping patterns, Key map, DTO, Identity map, Table data gateway, Data Mapper, etc.

- **Sequence diagrams** - **Three diagrams** for key operations (show object cooperation) for the implemented UC functionality.

Artifact 6 – Component and deployment diagram

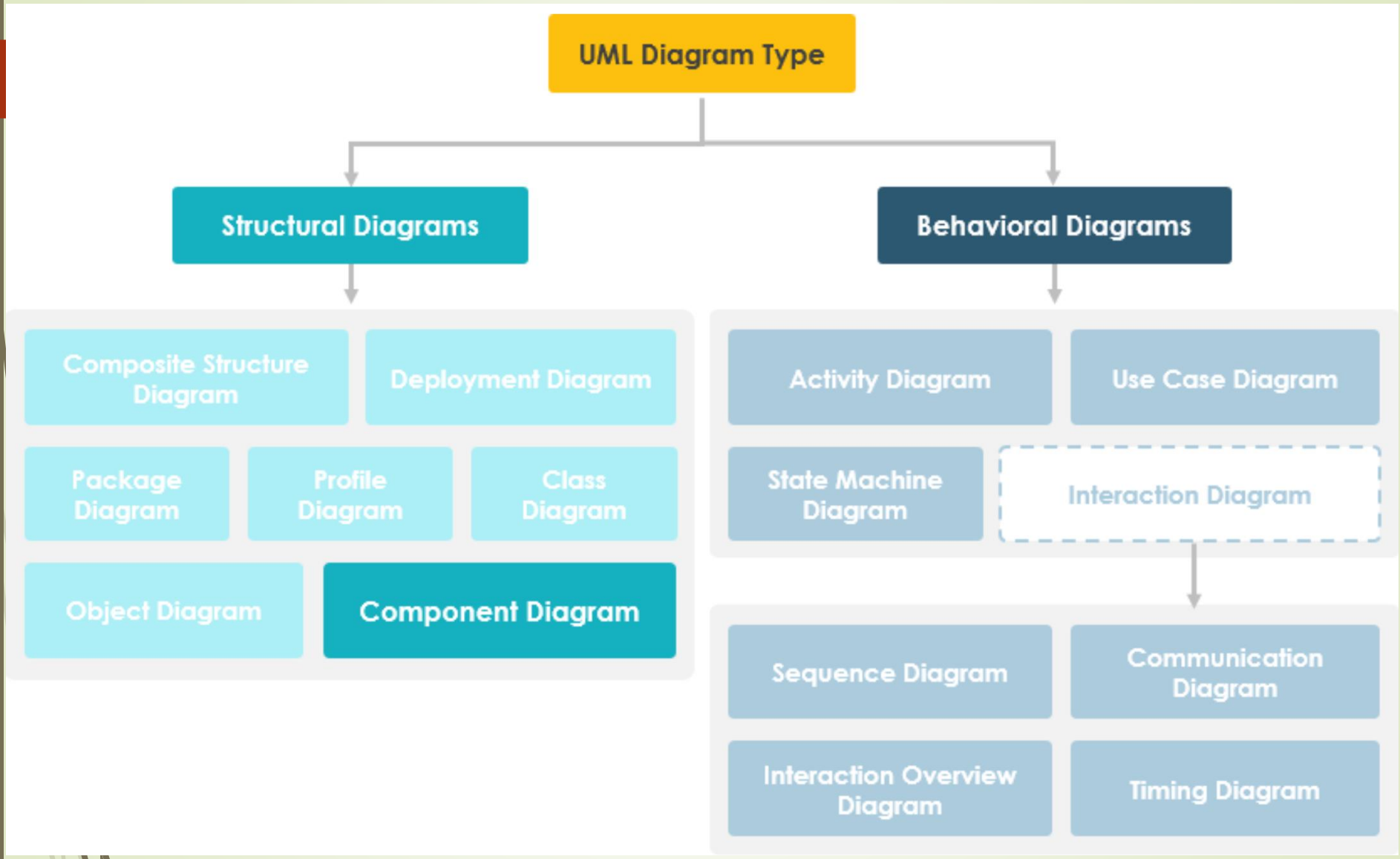
- **Component diagram** - shows the IS divided into severals logical component

- **Deployment diagram** - shows the execution architecture of a system

Component diagram

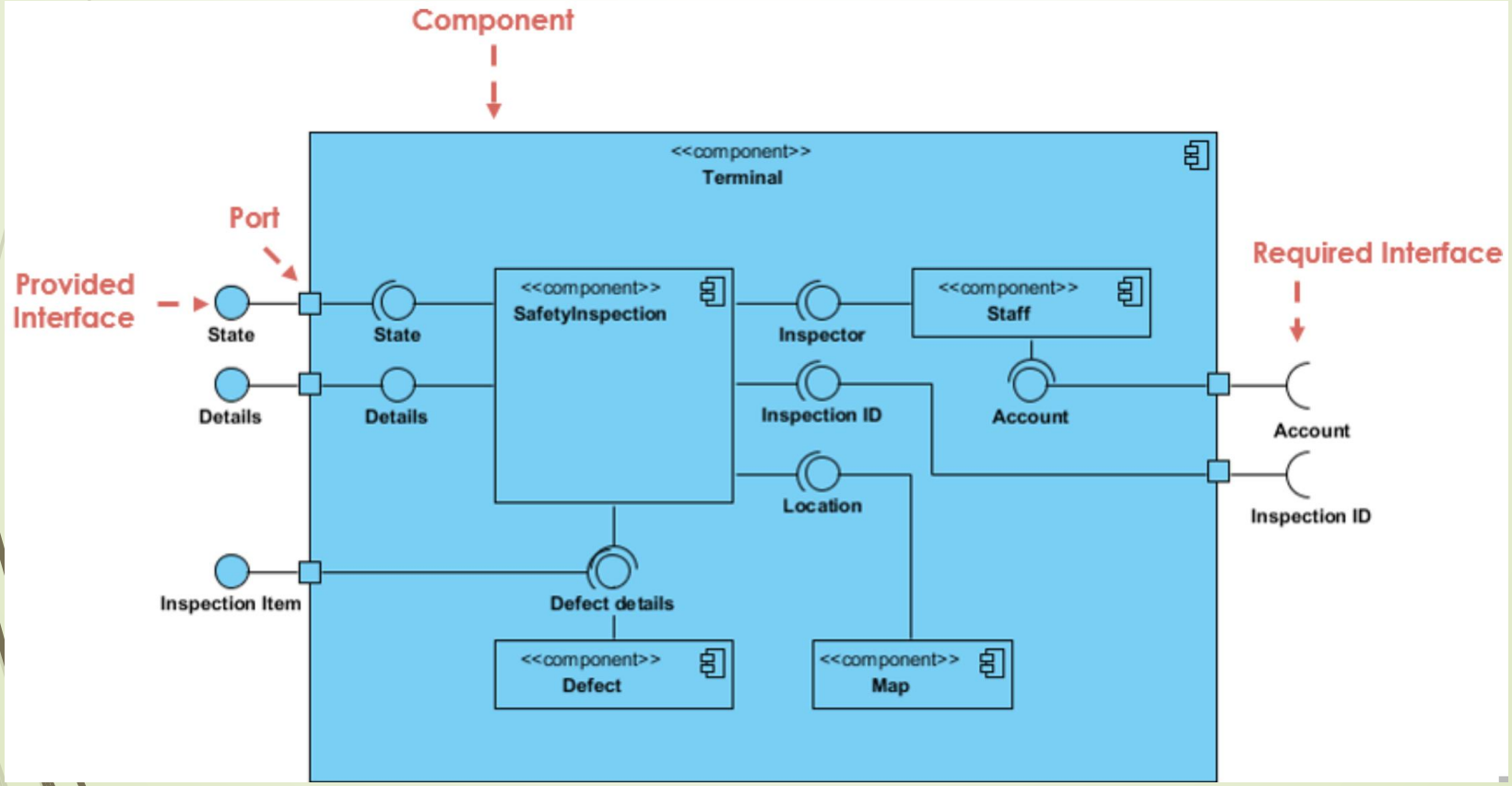
Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering.

Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.


UML Diagram Type
Structural Diagrams
Behavioral Diagrams
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram
Class Diagram
Object Diagram
Component Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Interaction Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram

Component diagram

- A component diagram breaks down the actual system under development into various high levels of functionality (**component**).
- Each component is responsible for one clear aim within the entire system and only interacts with other essential elements on a need-to-know basis.

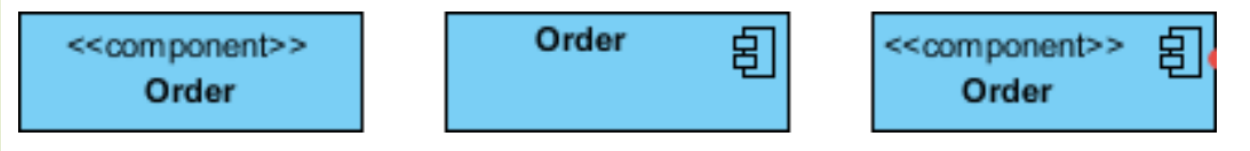


Component diagram

Component

A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. In UML 2, a component is drawn as a rectangle with optional compartments stacked vertically. A high-level, abstracted view of a component in UML 2 can be modeled as:

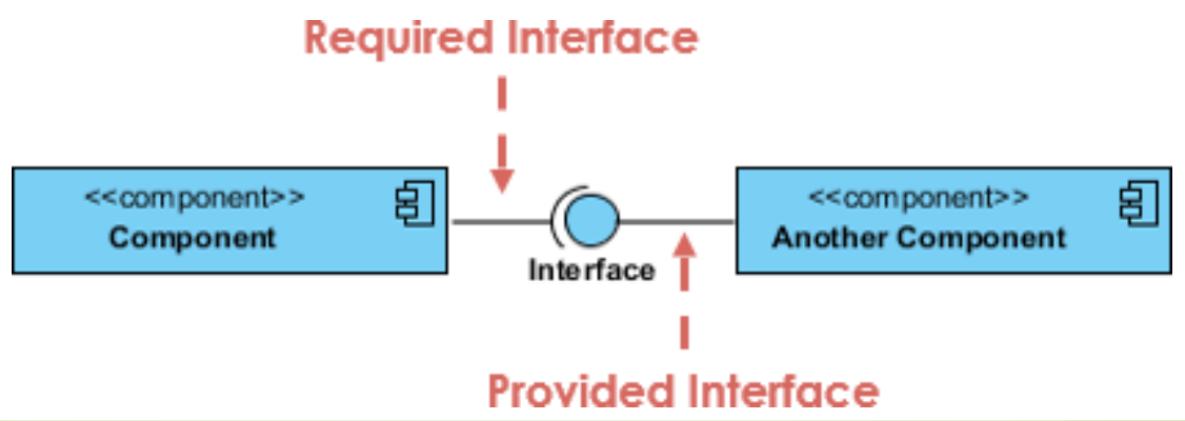
1. A rectangle with the component's name
2. A rectangle with the component icon
3. A rectangle with the stereotype text and/or icon



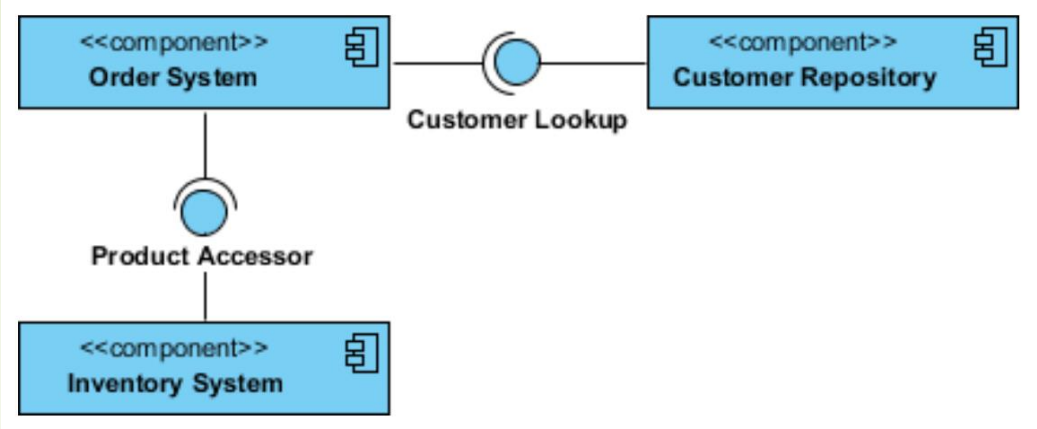
Component diagram

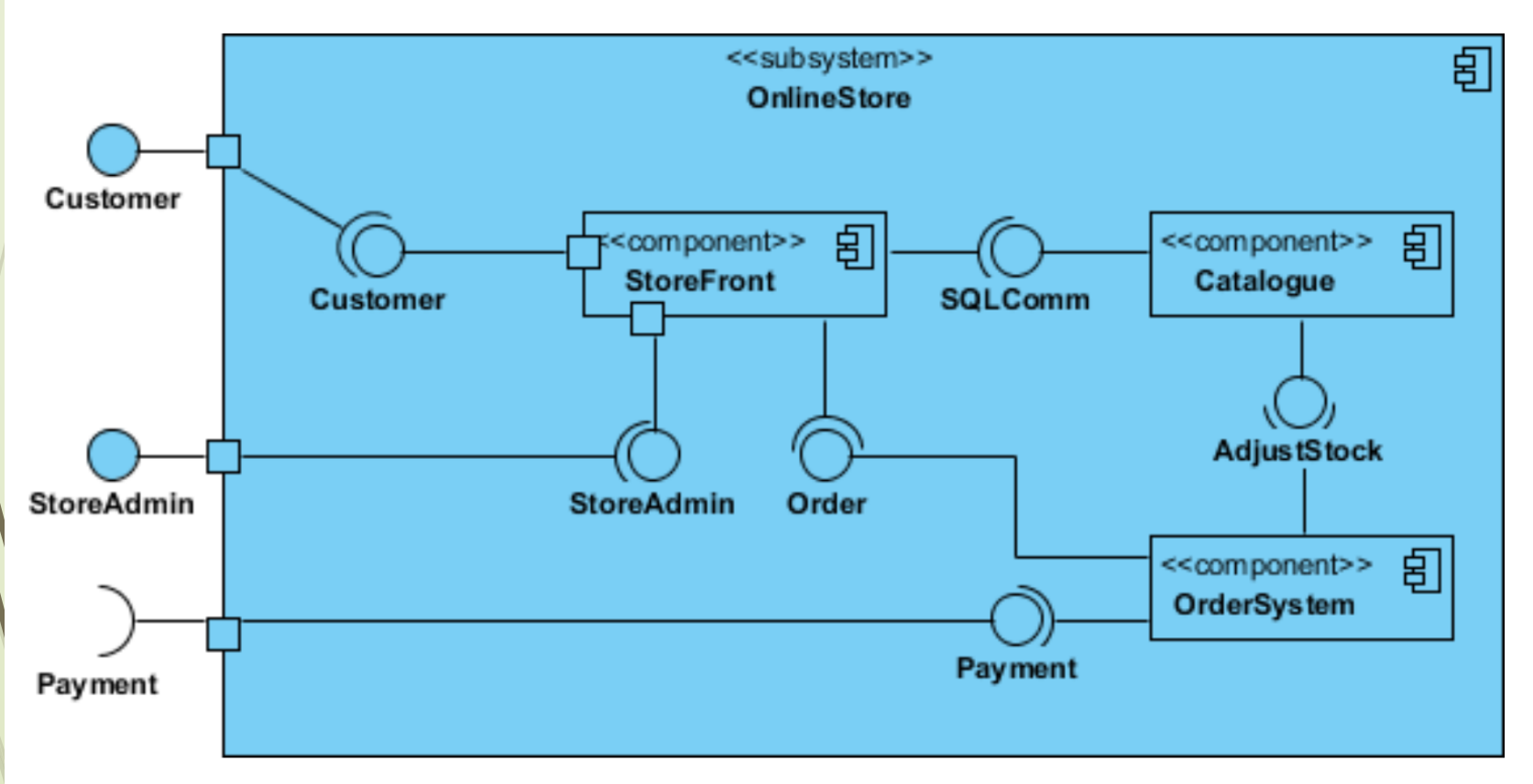
Interface

- **Provided interface** symbols with a complete circle at their end represent an interface that the component provides - this "lollipop" symbol is shorthand for a realization relationship of an interface classifier.
- **Required Interface** symbols with only a half circle at their end (a.k.a. sockets) represent an interface that the component requires (in both cases, the interface's name is placed near the interface symbol itself).



Component diagram

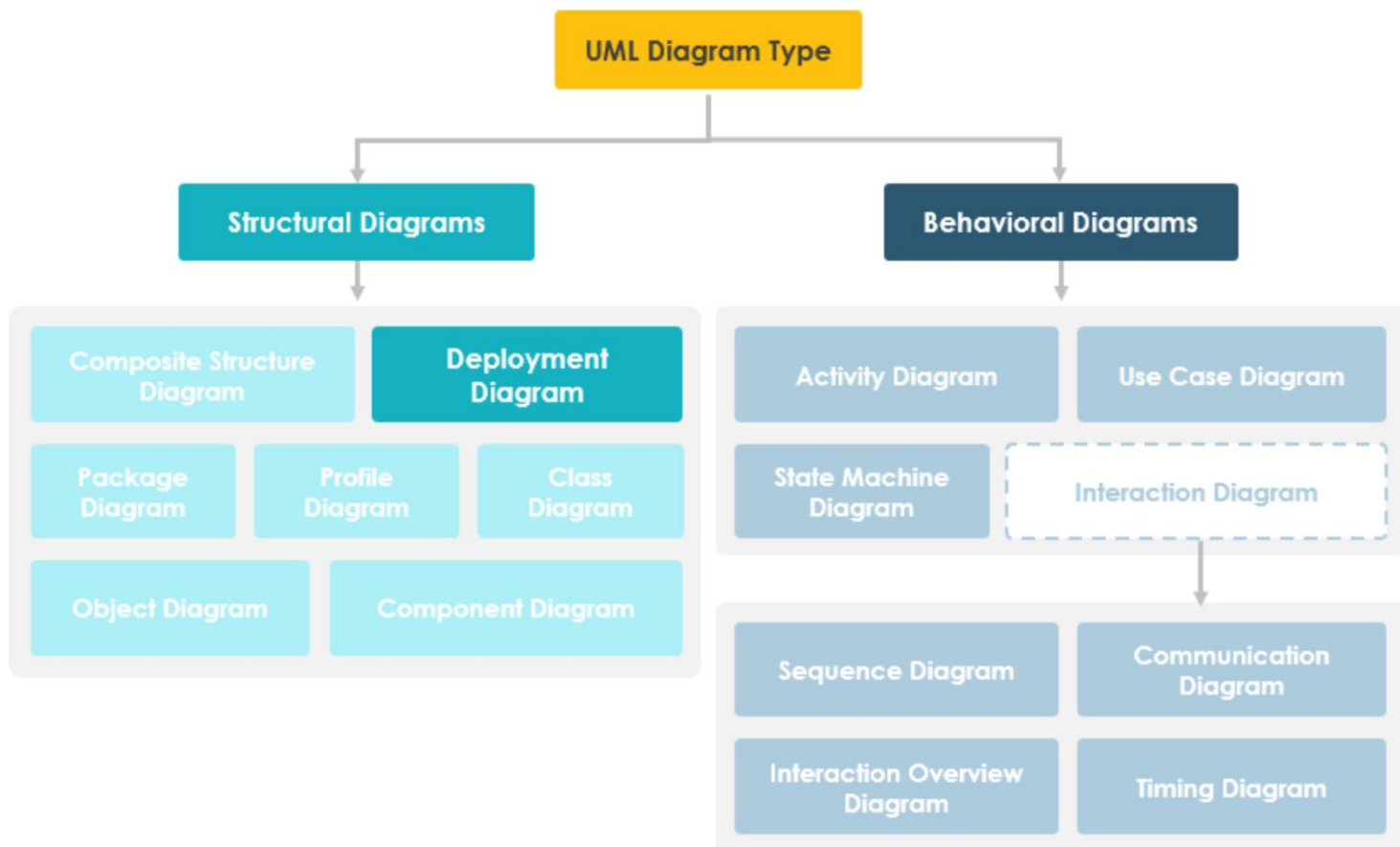




Deployment diagram

Deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them.

Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

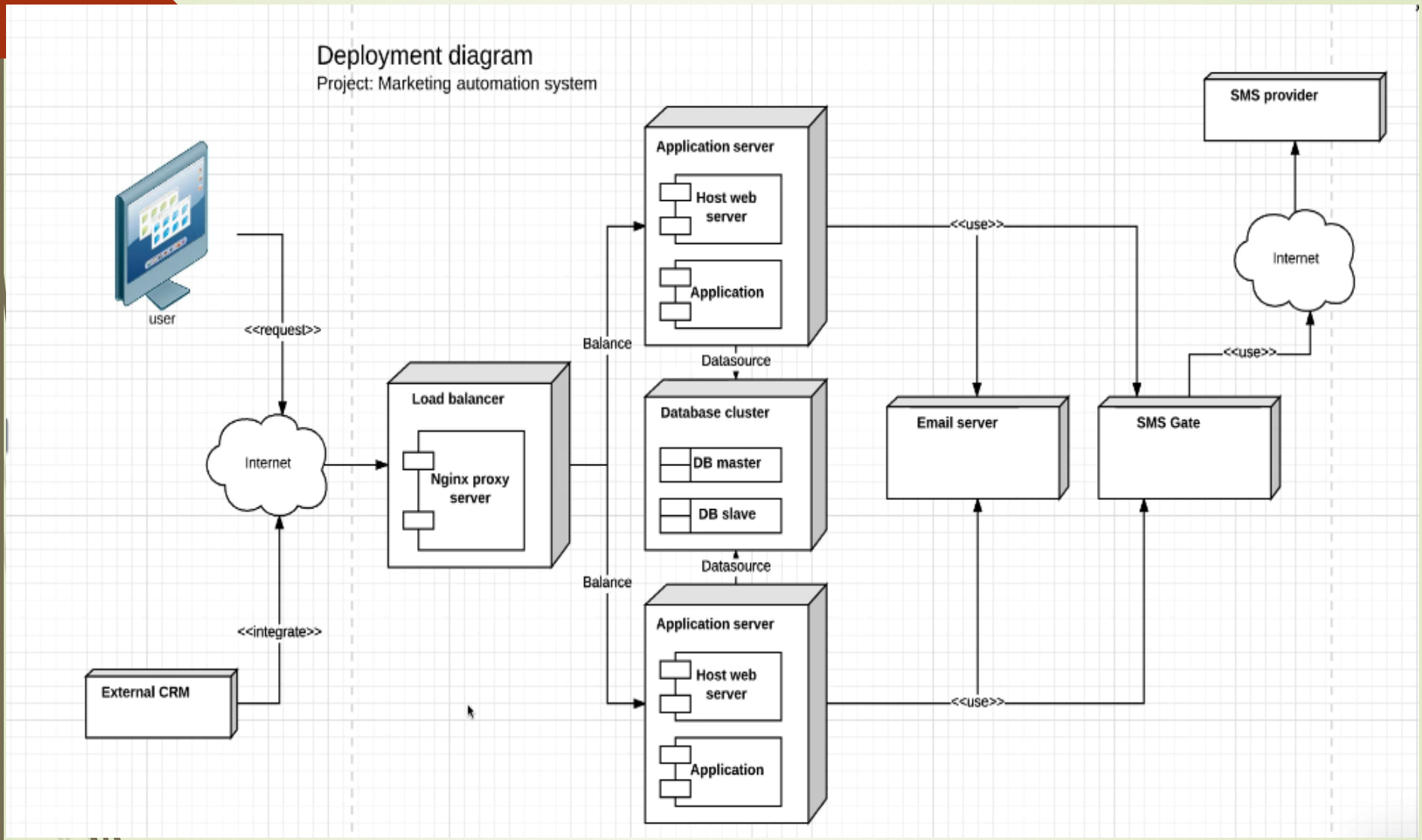


Deployment diagram

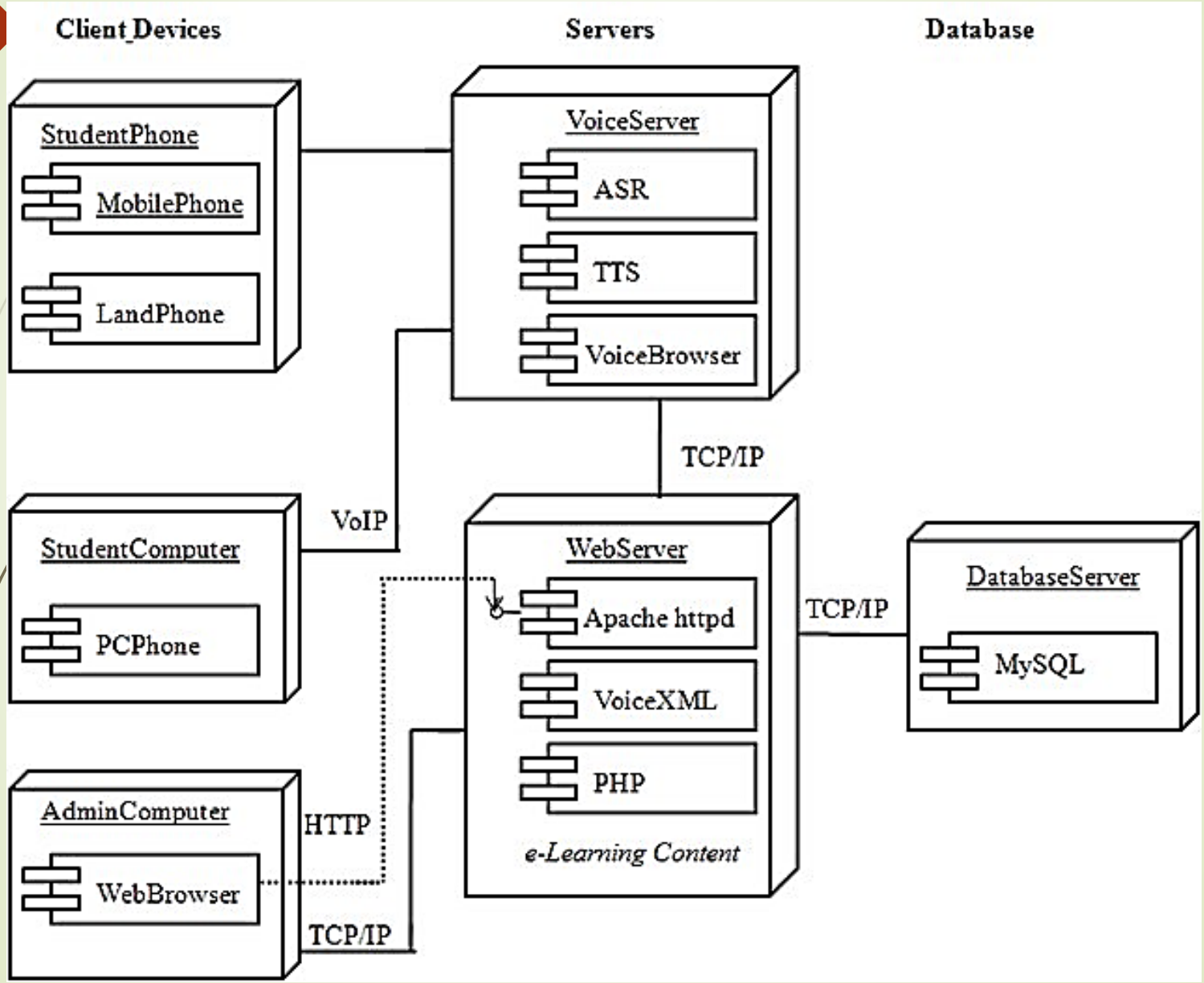
Deployment diagrams are important for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering.

- They show the structure of the run-time system
- They capture the hardware that will be used to implement the system and the links between different items of hardware.
- They model physical hardware elements and the communication paths between them
- They can be used to plan the architecture of a system.
- They are also useful for Document the deployment of software components or nodes

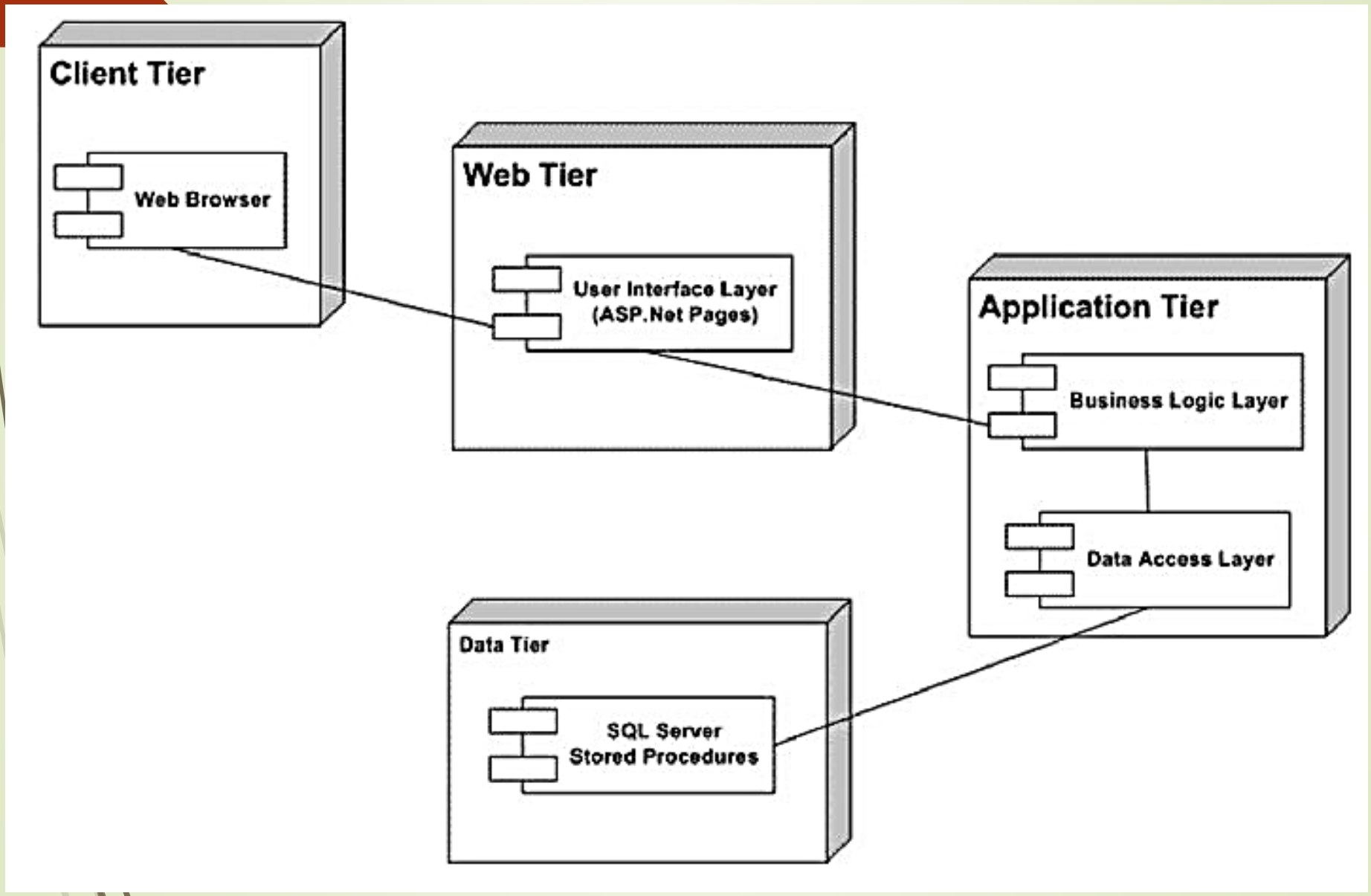
Deployment diagram



Deployment diagram



Deployment diagram



Discussion