# Object Oriented Design

# Objectives

- Understand the transition from analysis to design.
- Understand the use of factoring, partitions, and layers.
- Be able to create package diagrams.
- Be familiar with the custom, packaged, and outsource design alternatives.

# Factoring

- is the process of separating out a *module* into a standalone module in and of itself. The new module can be a new *class* or a new *method*. For example, when reviewing a set of classes, it may be discovered that they have a similar set of attributes and methods.
- As such, it may make sense to factor out the similarities into a separate class. Depending on whether the new class should be in a superclass relationship to the existing classes or not, the new class can be related to the existing classes through *generalization* (*A-Kind- Of*) or possibly through *aggregation* (*Has-Parts*) relationship.

# Partition

- A partition is the object-oriented equivalent of a subsystem, 2 where a subsystem is a decomposition of a larger system into its component systems (e.g., an accounting information system could be functionally decomposed into an accounts payable system, an account receivable system, a payroll system, and so on).

# Layer

- A layer represents an element of the software architecture of the evolving system.
- A layer that focuses on one layer in the evolving software architecture includes the ***problem domain layer***.
- There should be a layer for each of the different elements of the system environment (e.g., system architecture, user interface, and data access and management).

# Types of Layer

- **Foundation**
  - It contains classes that are necessary for any object-oriented application to exist.
  - They include classes that represent fundamental data types (e.g., integers, real numbers, characters, and strings), classes that represent fundamental data structures (sometimes referred to as container classes; e.g., lists, trees, graphs, sets, stacks, and queues), and classes that representuseful abstractions (sometimes referred to as utility classes; e.g., date, time, and money).

# Types of Layer

- **Physical Architecture**
  - The *physical architecture layer* addresses how the software will execute on specific computers and networks.
  - As such, this layer includes classes that deal with communication between the software and the computer's operating system and the network.
  - For example, classes that address how to interact with the various ports on a specific computer would be included in this layer. This layer also includes classes that would interact with so-called *middleware* applications

# Types of Layer

- Issues in Physical Architecture
  - These design issues include the choice of a computing or network architecture (such as the various client-server architectures), the actual design of a network, hardware and server software specification, global/international issues (such as multilingual requirements), and security issues.

# Types of Layer

- **Human Computer Interaction**
  - The primary purpose of this layer is to keep the specific user interface implementation separate from the problem domain classes.
  - Typical classes found on this layer include classes that can be used to represent buttons, windows, text fields, scroll bars, check boxes, drop-down lists, and many other classes that represent user interface elements

# Types of Layer

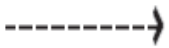
- *Data management layer*
  - addresses the issues involving the persistence of the objects contained in the system. The types of classes that appear in this layer deal with how objects can be stored and retrieved.
  - Some of the issues related to this layer include choice of the storage format (such as relational, object/relational, and object databases) and storage optimization (such as clustering and indexing).

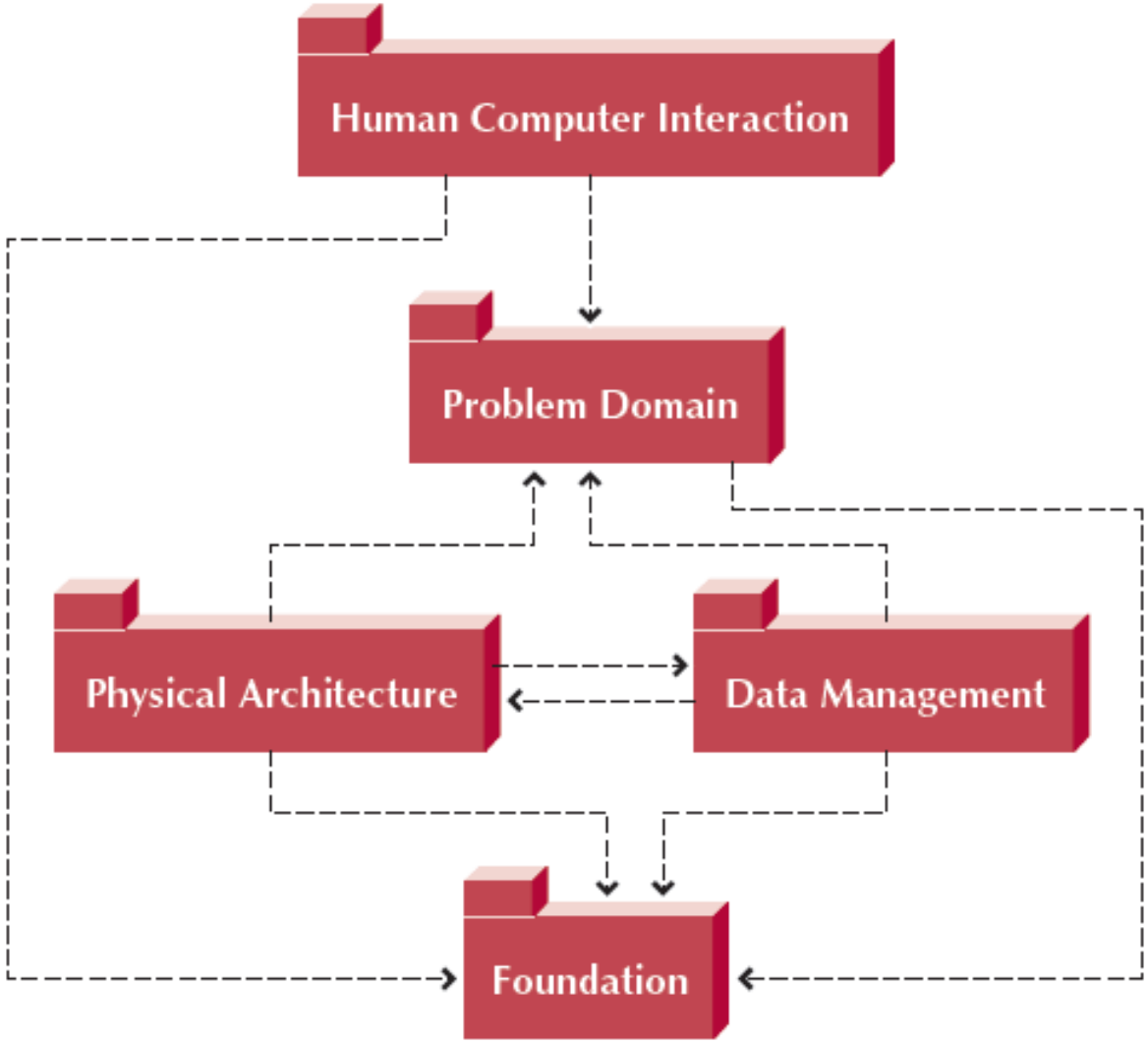
# Types of Layer

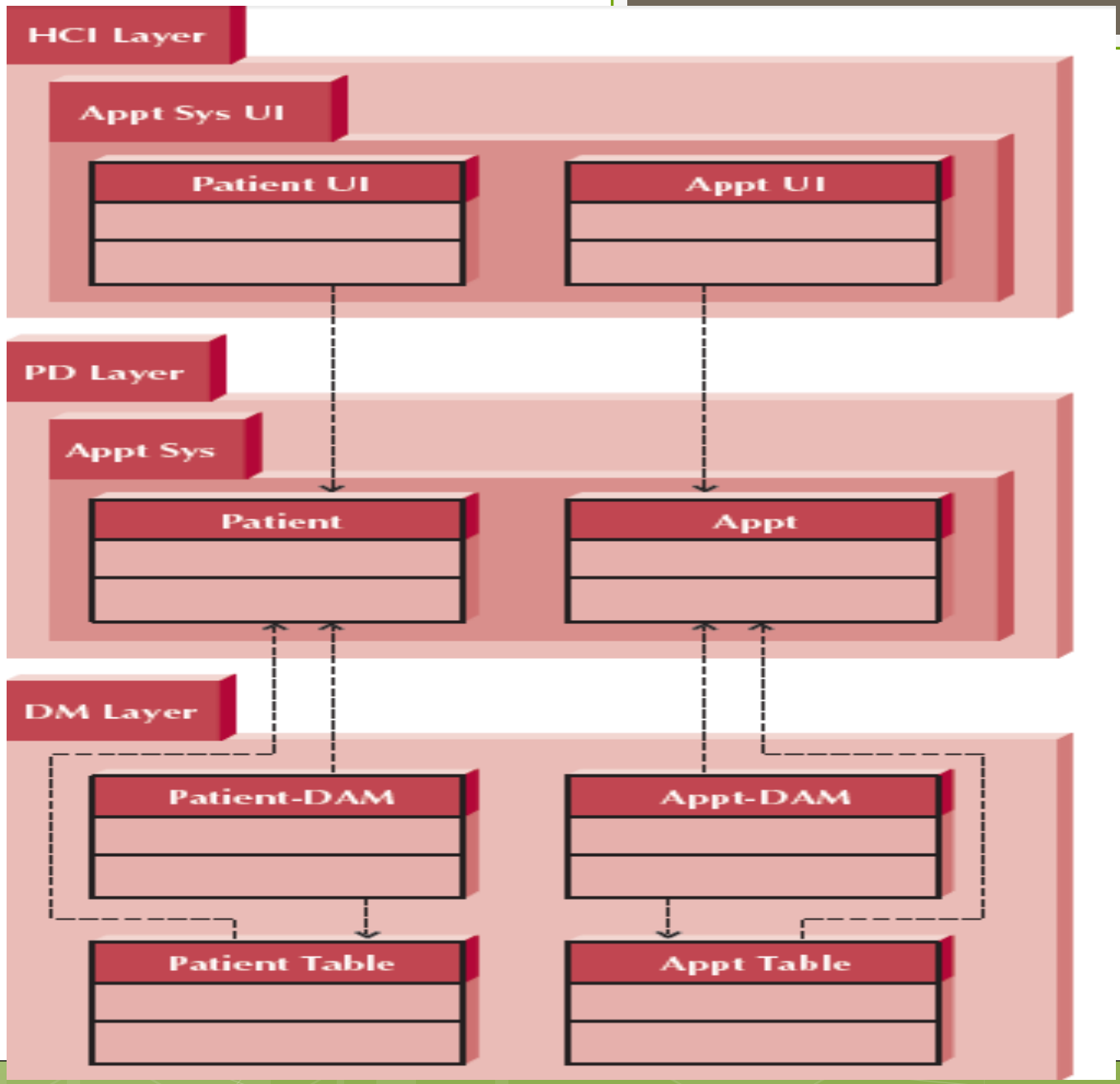
- **Problem Domain**
  - The previous layers dealt with classes that represent elements from the system environment and which will be added to the evolving system specification.

# UML Package Diagram

- A *package*
  - is a general construct that can be applied to any of the elements in UML models.
- Package diagram:
  - a diagram that is composed only of packages. A *package diagram* is effectively a class diagram that only shows packages.

| | |
|---|---|
| **A Package:**<br>■ A logical grouping of UML elements<br>■ Used to simplify UML diagrams by grouping related elements into a single higher-level element |  |
| **A Dependency Relationship:**<br>■ Represents a dependency between packages, i.e., if a package is changed, the dependent package also could have to be modified<br>■ The arrow is drawn from the dependent package toward the package on which it is dependent |  |


Human Computer Interaction
Problem Domain
Physical Architecture
Data Management
Foundation


HCI Layer
Appt Sys UI
Patient UI
Appt UI
PD Layer
Appt Sys
Patient
Appt
DM Layer
Patient-DAM
Appt-DAM
Patient Table
Appt Table

# Steps ion Creating Package Diagram

- Set the Context.
- Cluster classes together based on shared relationships.
- Model clustered classes as a package.
- Identify dependency relationships among packages.
- Place dependency relationships between packages.

# Example