VI SEMESTER

SOFTWARE ENGINEERING
&
PROJECT MANAGEMENT
21CST601

Dr. PRERANA CHAITHRA

UNIT 3

1. DESIGN CONCEPTS
2. ARCHITECTURAL DESIGN
3. COMPONENT-LEVEL DESIGN

UNIT 3

- **Design Concepts :**
  - Design within the Context of Software Engineering
  - The Design Process
  - Design Concepts
  - The Design Model

- **Architectural Design :**
  - Software Architecture
  - Definition of software architecture
  - Architectural Genres
  - Architectural Styles
  - Architectural Design

UNIT 3

(Contd..)

- **Component-level Design :**
  - What Is a Component?
  - Designing Class-Based Components
  - Conducting Component-Level Design
  - Designing Traditional Components and Component-Based Development

Unit 3.1

Design Concepts

UNIT 3.1

- **Design Concepts :**
  - Design within the Context of Software Engineering
  - The Design Process
  - Design Concepts
  - The Design Model

Chapter 12

Design Concepts

Design Concepts

- **Software design**
- consists of the set of
  - Principles
  - Concepts
  - practices
- that lead to the development of a high-quality system or product
- Design is the place where quality is fostered in software engineering.

3.1.1

Design within the Context of Software Engineering

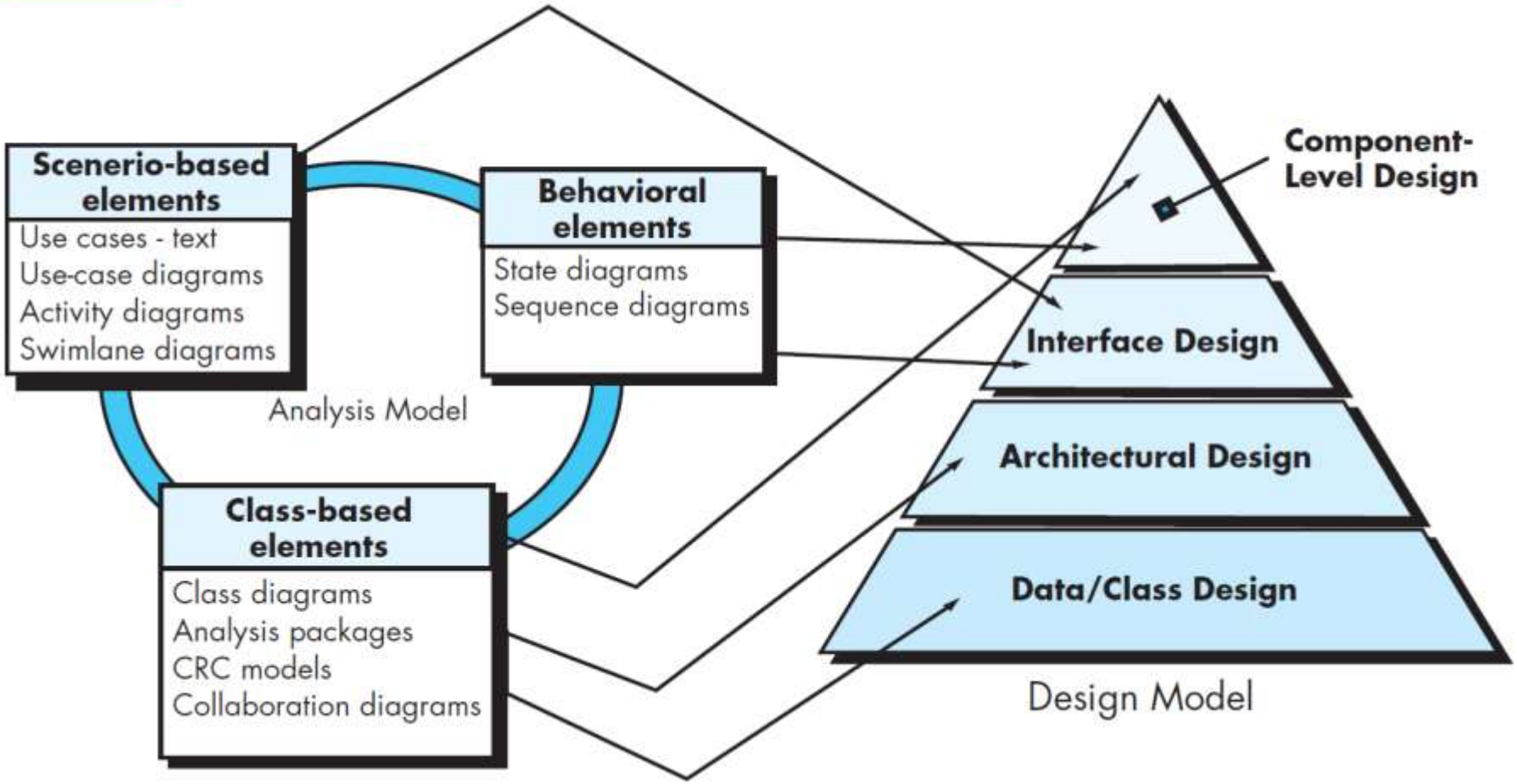
Design within the Context of Software Engineering

- Each of the elements of the requirements model provides information that is necessary to create the four design models required for a complete specification of design

- **Design Models**
  - Data/Class Design
  - Architectural Design
  - Interface Design
  - Component-level Design

Design within the Context of Software Engineering (Contd..)

FIGURE 12.1 Translating the requirements model into the design model



Design within the Context of Software Engineering (Contd..)

- **Data/Class Design**

  transforms class models into
  - design class realizations
  - requisite data structures required to implement the software

- **Architectural Design**

  defines the relationship between major structural elements of the software, the architectural styles and patterns

- **Interface Design**

  describes how the software communicates with systems that interoperate with it, and with humans who use it

- **Component-level Design**

  transforms structural elements of the software architecture into a procedural description of software components

3.1.2

The Design Process

The Design Process

- **Software design**

  is an iterative process through which requirements are translated into a "blueprint" for constructing the software

The Design Process (Contd..)
Software Quality Guidelines and Attributes

- **McGlaughlin suggests three characteristics that serve as a guide for the evaluation of a good design**
  - The design should implement all of the explicit requirements contained in the requirements model, and it must accommodate all of the implicit requirements desired by stakeholders.
  - The design should be a readable, understandable guide for those who generate code and for those who test and subsequently support the software.
  - The design should provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.

The Design Process (Contd..)
Quality Guidelines

1. A design should exhibit an architecture that

   (1) has been created using recognizable architectural styles or patterns,

   (2) is composed of components that exhibit good design characteristics

   (3) can be implemented in an evolutionary fashion, thereby facilitating implementation and testing.

2. A design should be modular; that is, the software should be logically partitioned into elements or subsystems.

3. A design should contain distinct representations of data, architecture, interfaces, and components.

4. A design should lead to data structures that are appropriate for the classes to be implemented and are drawn from recognizable data patterns.

The Design Process (Contd..)
Quality Guidelines

5. A design should lead to components that exhibit independent functional characteristics.

6. A design should lead to interfaces that reduce the complexity of connections between components and with the external environment.

7. A design should be derived using a repeatable method that is driven by information obtained during software requirements analysis.

8. A design should be represented using a notation that effectively communicates its meaning.

The Design Process
Quality Attributes

- Hewlett-Packard developed a set of software quality attributes that has been given the acronym **FURPS**
  - **Functionality**
  - **usability**
  - **Reliability**
  - **Performance**
  - **Supportability**
- The FURPS quality attributes represent a target for all software design:

The Design Process (Contd..)
Quality Attributes

- **Functionality**
  - is assessed by evaluating the feature set and capabilities of the program
  - the generality of the functions that are delivered
  - The security of the overall system

- **Usability**
  - is assessed by considering
    - human factors
    - overall aesthetics
    - Consistency
    - documentation

The Design Process (Contd..)
Quality Attributes

- **Reliability**
  - is evaluated by
    - measuring the frequency and severity of failure
    - the accuracy of output results
    - the mean-time-to-failure (MTTF)
    - the ability to recover from failure
    - the predictability of the program

- **Performance**
  - is measured using
    - processing speed
    - response time
    - Resource
    - Consumption
    - Throughput
    - efficiency

The Design Process (Contd..)
Quality Attributes

- **Supportability**
- combines extensibility, adaptability, and serviceability.
- These three attributes represent a more common term, maintainability
- in addition,
  - Testability
  - Compatibility
  - configurability,
  - the ease with which a system can be installed
  - the ease with which problems can be localized.

The Design Process
The Evolution of Software Design

- **The evolution of software design is a continuing process** that has now spanned more than six decades.
- Early design for the development of modular programs and methods for refining software structures in a top-down "structured" manner
- Newer design approaches proposed an object-oriented approach to design derivation.
- More recent emphasis in software design has been on software architecture and the design patterns that can be used to implement software architectures and lower levels of design abstractions
- Growing emphasis on aspect-oriented methods model-driven development and test-driven development emphasize techniques for achieving more effective modularity and architectural structure in the designs that are created

3.1.3
Design Concepts

Design Concepts

- **Fundamental Software Design Concepts**
  - Abstraction
  - Architecture
  - Patterns
  - Separation of Concerns
  - Modularity
  - Information Hiding
  - Functional Independence
  - Refinement
  - Aspects
  - Refactoring
  - Object-Oriented Design Concepts
  - Design Classes
  - Dependency Inversion
  - Design for Test

Design Concepts (Contd..)

- **Abstraction**
- Types of Abstraction
  - Procedural Abstraction
  - Data Abstraction
- **Procedural Abstraction** refers to a sequence of instructions that have a specific and limited function
- **Data Abstraction** is a named collection of data that describes a data object

Design Concepts (Contd..)

Architecture

- Software architecture alludes to "the overall structure of the software and the ways in which that structure provides conceptual integrity for a system
- Set of properties specified as part of an architectural design
  - Structural properties
  - Extra-functional properties
  - Families of related systems
- Architectural design can be represented using one or more models
  - Structural models
  - Framework models
  - Dynamic models
  - Process models
  - functional models
- A number of different architectural description languages (ADLs) have been developed to represent these models

Design Concepts (Contd..)

Patterns

- A *design pattern* or a pattern is a named nugget of insight which conveys the essence of a proven solution to a recurring problem within a certain context amidst competing concerns
- Design pattern provides a description that enables a designer to determine

  (1) whether the pattern is applicable to the current work

  (2) whether the pattern can be reused

  (3) whether the pattern can serve as a guide for developing a similar, but functionally or structurally different pattern.

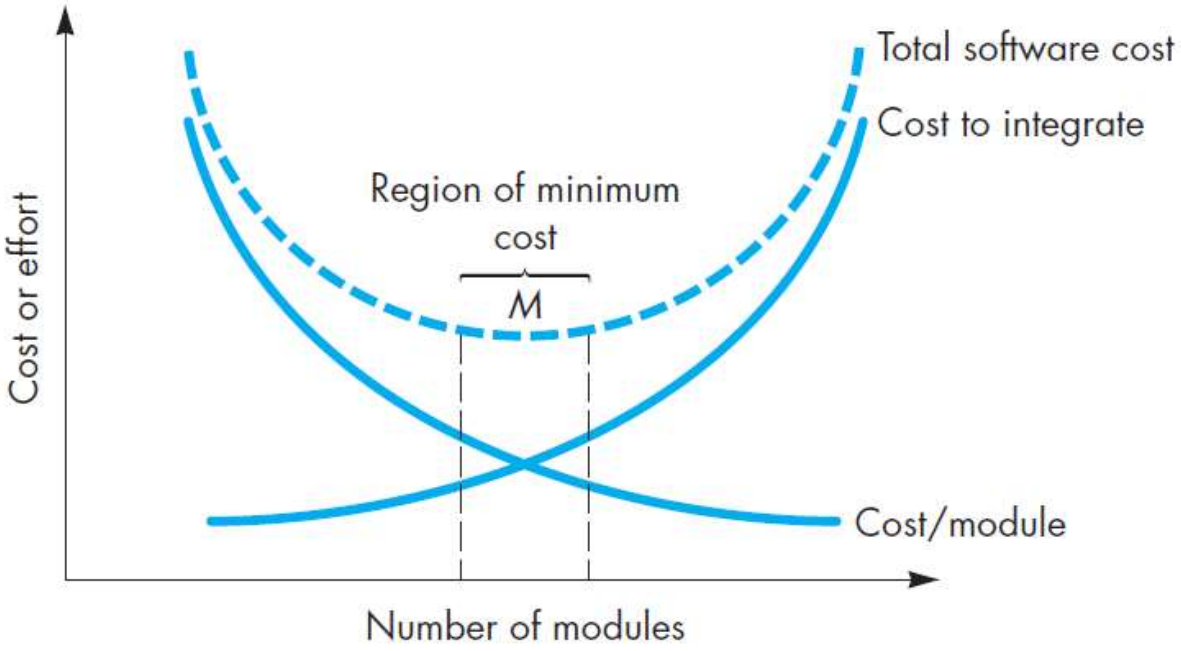
Design Concepts (Contd..)

- **Separation of Concerns**
  - is a design concept
  - suggests that any complex problem can be more easily handled if it is subdivided into pieces that can each be solved and/or optimized independently
  - Concern is a feature or behavior that is specified as part of the requirements model for the software

Design Concepts (Contd..)

Modularity

- Software is divided into separately named and addressable components, called *modules,* that are integrated to satisfy problem requirements



Design Concepts (Contd..)

- **Information Hiding**
  - modules should be specified and designed so that information (algorithms and data) contained within a module is inaccessible to other modules that have no need for such information.

Design Concepts (Contd..)

- **Functional Independence**
  - is achieved by designing software so that each module addresses a specific subset of requirements and has a simple interface when viewed from other parts of the program structure
  - Independence is assessed using two qualitative criteria:
    - Cohesion & Coupling
  - Cohesion
    - indication of the relative functional strength of a module
    - extension of the information-hiding concept
  - Coupling
    - indication of the relative interdependence among modules
    - interconnection among modules in a software structure
- Ripple Effect is caused when errors occur at one location and propagate throughout a system

Design Concepts (Contd..)

- **Refinement**
  - Refinement is a process of elaboration
  - An application is developed by successively refining levels of procedural detail
  - Begin with a statement of function that is defined at a high level of abstraction.
  - then elaborate on the original statement, providing more and more detail as each successive refinement (elaboration) occurs.
  - Stepwise refinement is a top-down design strategy

Design Concepts (Contd..)

- **Aspects**
  - An aspect is a representation of a crosscutting concern
  - **Crosscuts**
  - Consider two requirements, A and B.
  - Requirement A crosscuts requirement B "if a software decomposition [refinement] has been chosen in which
  - B cannot be satisfied without taking A into account

Design Concepts (Contd..)

Refactoring

- is a reorganization technique that simplifies the design (or code) of a component without changing its function or behavior
- Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure

Object-Oriented Design Concepts

- The Object-oriented (OO) paradigm is widely used in modern software engineering.
- OO design concepts consists of classes and objects, inheritance, messages, polymorphism, etc.

Design Concepts (Contd..)

Design Classes

- set of design classes refine the analysis classes by providing design detail that will enable the classes to be implemented, and implement a software infrastructure that supports the business solution.
- **Five different types of design classes**
- User interface classes define all abstractions that are necessary for human-computer interaction (HCI)
- Business domain classes identify the attributes and services (methods) that are required more analysis classes.
- Process classes implement lower-level business abstractions required to fully manage the business domain classes.
- Persistent classes represent data stores (e.g., a database) that will persist beyond the execution of the software.
- System classes implement software management and control functions that enable the system to operate and communicate within its computing environment and with the outside world

Design Concepts (Contd..)

- **Four characteristics of a well-formed design class**
- Complete and sufficient A design class should be the complete encapsulation of all attributes and methods that can reasonably be expected to exist for the class.
- Primitiveness Methods associated with a design class should be focused on accomplishing one service for the class. Once the service has been implemented with a method, the class should not provide another way to accomplish the same thing
- High cohesion A cohesive design class has a small, focused set of responsibilities and single-mindedly applies attributes and methods to implement those responsibilities
- Low coupling Within the design model, it is necessary for design classes to collaborate with one another. However, collaboration should be kept to an acceptable minimum.

Design Concepts (Contd..)

- **Dependency Inversion**
  - High-level modules (classes) should not depend [directly] upon low-level modules.
  - Both should depend on abstractions.
  - Abstractions should not depend on details.
  - Details should depend on abstractions.

Design Concepts (Contd..)

Design for Test

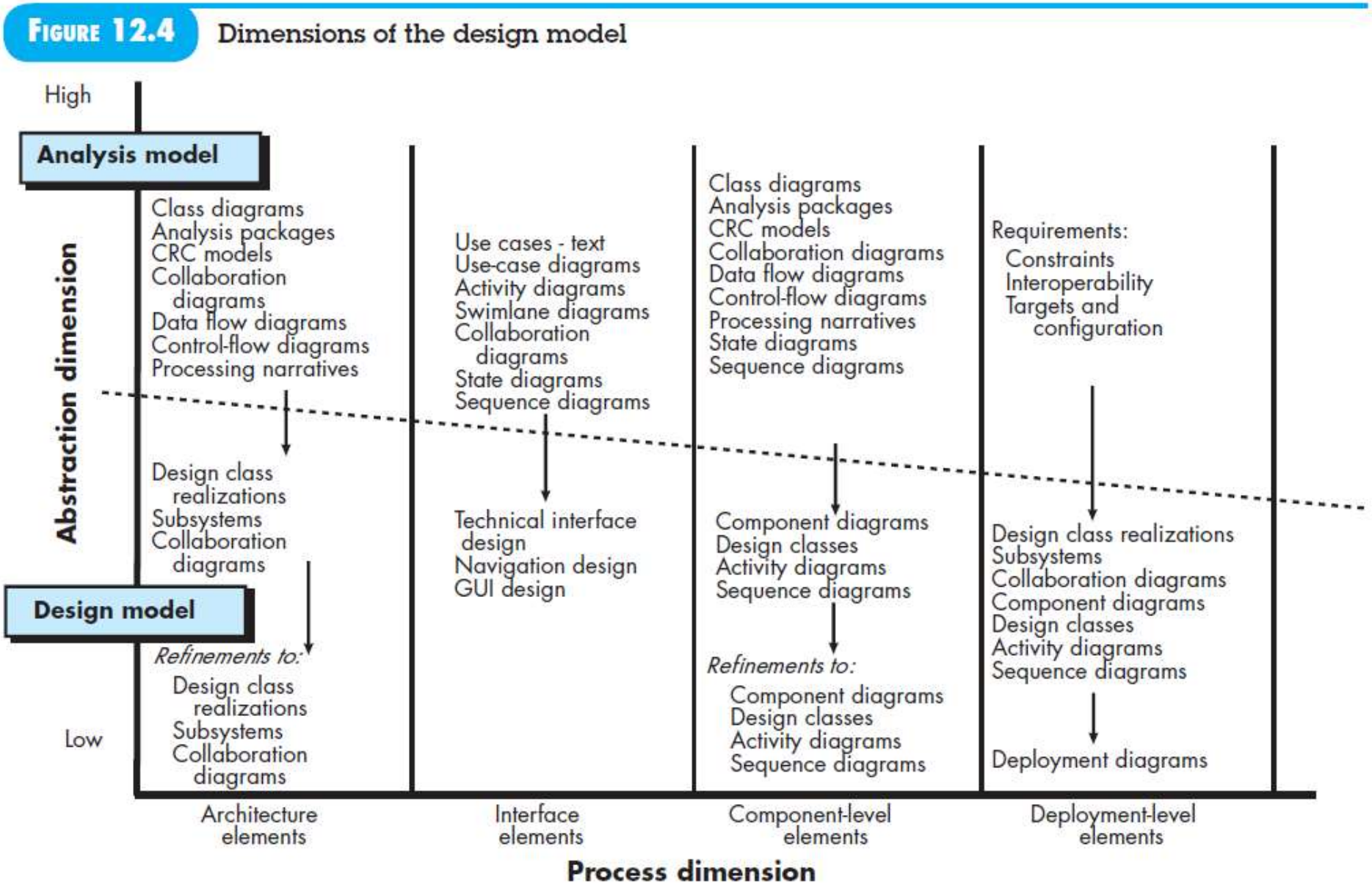
- There is an ongoing debate whether software design or test case design should come first.
- According to test-driven development (TDD) tests must be written before implementing any other code.
- But if design comes first, then the design (and code) must be developed with seams —locations in the detailed design where test code can be inserted that probes the state of the running software"
- isolate code under test from its production environment so that you can exercise it in a controlled testing context

3.1.4
The Design Model

The Design Model

- **The design model can be viewed in two different dimensions**
  - Process Dimension indicates the evolution of the design model as design tasks are executed as part of the software process.
  - Abstraction Dimension represents the level of detail as each element of the analysis model is transformed into a design equivalent and then refined iteratively

The Design Model (Contd..)



The Design Model (Contd..)
Data Design Elements

- **Data Design or Data Architecting**

- creates a model of data and/or information that is represented
- at a high level of abstraction (the customer/user's view of data)
- This data model is then refined into progressively more implementation-specific representations that can be processed by the computer-based system.

The Design Model (Contd..)
Architectural Design Elements

- The architectural design for software is the equivalent to the floor plan of a house.

- **Architectural model is derived from three sources**

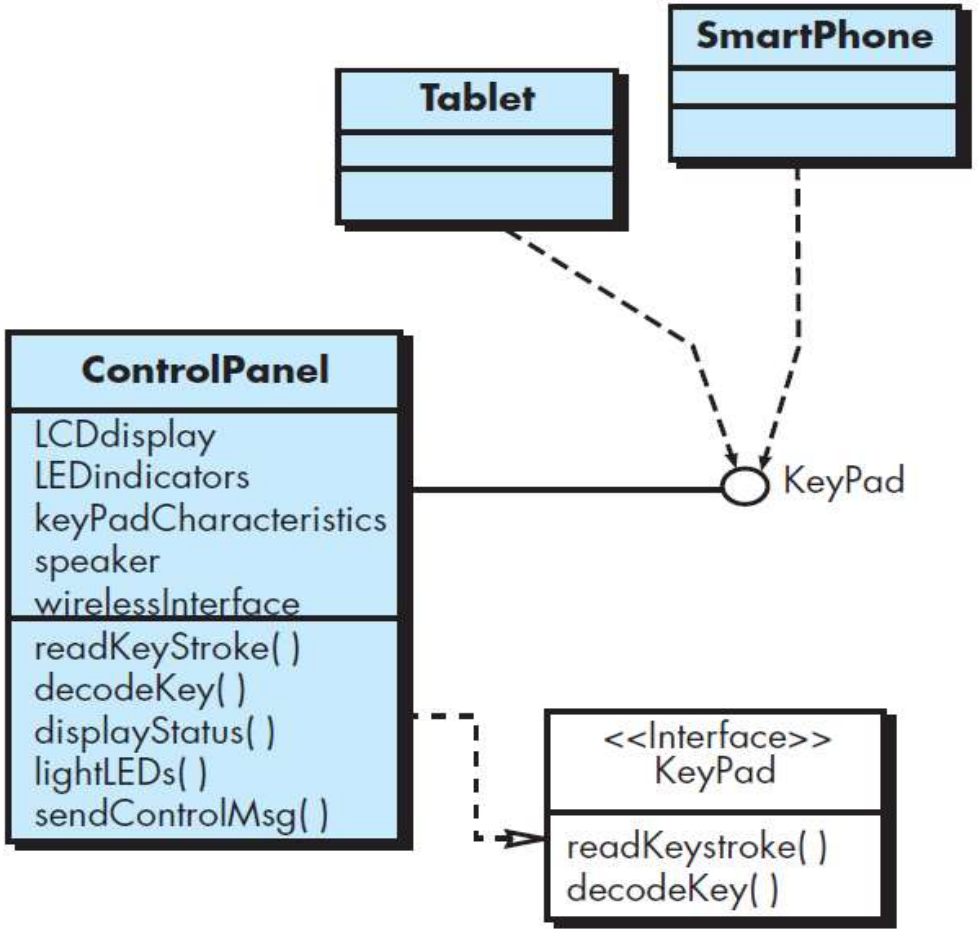
(1) Information about the application domain for the software to be built
(2) specific requirements model elements such as use cases or analysis classes, their relationships and collaborations for the problem at hand
(3) the availability of architectural styles

The Design Model (Contd..)
Interface Design Elements

- The **Interface Design Elements** for software depict information flows into and out of a system and how it is communicated among the components defined as part of the architecture.
- Three important elements of interface design:
  (1) the user interface (UI)
  (2) external interfaces to other systems, devices, networks, or other producers or consumers of information
  (3) internal interfaces between various design components.
- These interface design elements allow the software to communicate externally and enable internal communication and collaboration among the components that populate the software architecture.
- Usability design incorporates
  - aesthetic elements (e.g., layout, color, graphics, interaction mechanisms),
  - ergonomic elements (e.g., information layout and placement, metaphors, UI navigation)
  - technical elements (e.g., UI patterns, reusable components)

The Design Model (Contd..)
Interface Design Elements

- Interface representation for ControlPanel

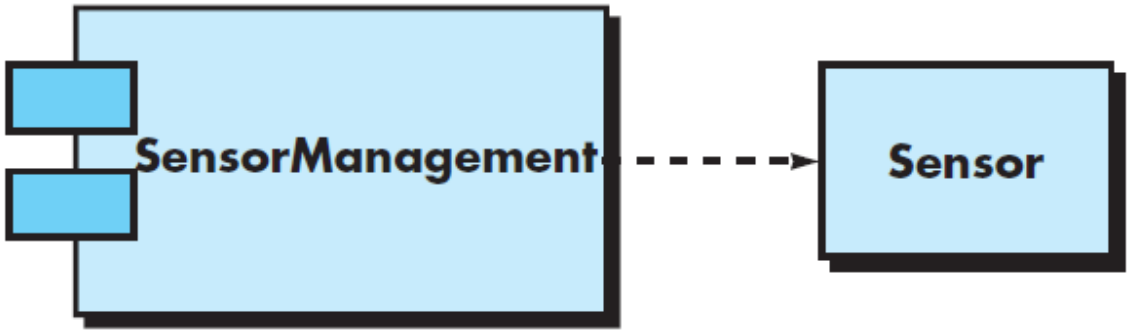


The Design Model (Contd..)
Component-Level Design Elements

- The **Component-level Design** for software fully describes the internal detail of each software component.

- To accomplish this, the component-level design defines data structures for all local data objects and algorithmic detail for all processing that occurs within a component and an interface that allows access to all component operations (behaviors).

- **UML component diagram**



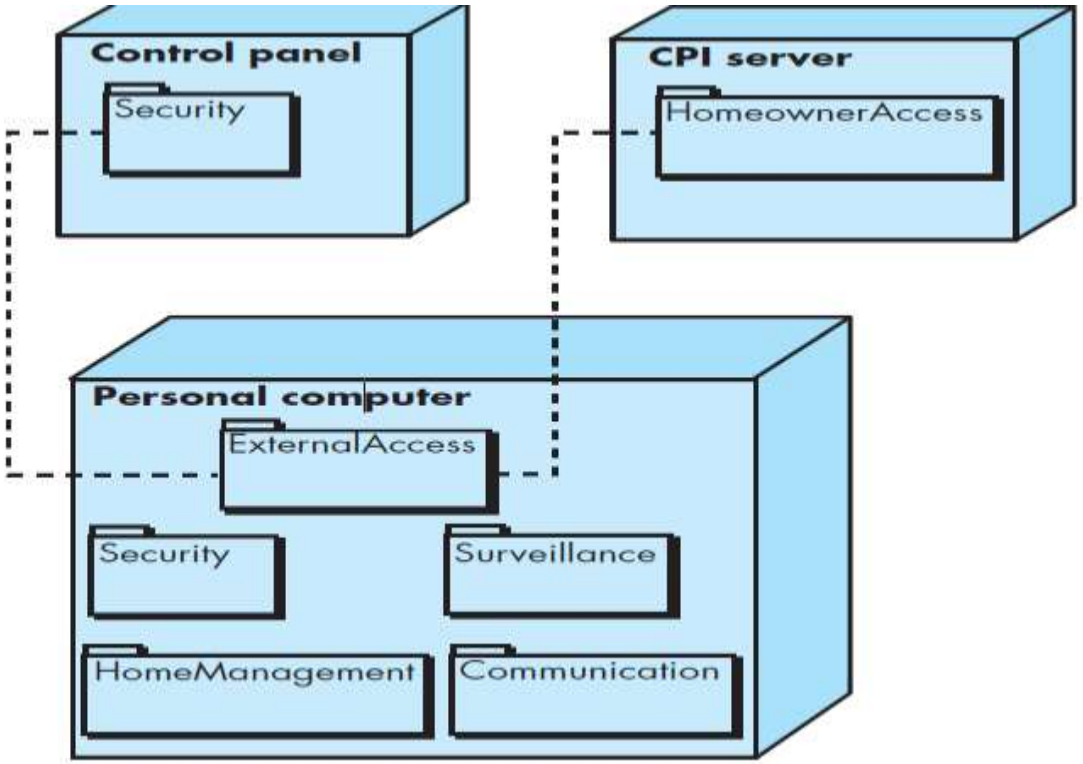
The Design Model (Contd..)
Component-Level Design Elements

- **The design details of a component can be modeled at many different levels of abstraction**

- A UML activity diagram can be used to represent processing logic.
- Detailed procedural flow for a component can be represented using either pseudocode or some other diagrammatic form (e.g., flowchart or box diagram)
- Algorithmic structure follows the rules established for structured programming (i.e., a set of constrained procedural constructs).
- Data structures, selected based on the nature of the data objects to be processed, are usually modeled using pseudocode or the programming language to be used for implementation

The Design Model (Contd..)
Deployment-Level Design Elements

- **Deployment-level design elements** indicate how software functionality and subsystems will be allocated within the physical computing environment that will support the software.
- During design, a UML deployment diagram is developed and then refined
- **UML deployment diagram**



THANK YOU