

# CS3773 Software Engineering

## Lecture 22 Software Design Method

## Software Design Process

- Software design is usually created in two stages
  - Architecture design
  - Detail design (component-level design)
    - Design patterns
    - Design classes
- Other design issues
  - Refactoring
  - Collaborative design
  - Model-driven design

2

UTSA CS3773

## Architecture Design

- Software architecture is derived from three sources
  - Information about the application domain
  - Specific requirements model elements
  - Availability of architectural styles
- Advantages of explicitly designing and documenting a software architecture
  - Stakeholder communication
  - System analysis
  - Large-scale reuse
- Architecture affects system's performance, reliability, etc.

3

UTSA CS3773

## Architecture Design

- Software architecture represents
  - Logical structure of the system
  - Global data structure
  - Global control flow
- Software architecture produces an architecture design document
  - Components properties and their interfaces
  - Interactions of components: data and control flows
  - External interfaces: GUI, databases, networks, files

4

UTSA CS3773

## Interface Design

- User interface can be tricky to design
  - Physical and mental capacities of users
  - Short-term memory of users
  - Users making mistakes
  - Different interaction preference
- Careful user interface design is an essential part of the overall software design
  - User interface should be designed to match the skills, experience and expectations of its anticipated users
  - Good user interface design is critical for system dependability

5

UTSA CS3773

## Interface Design Issues

- Key elements of interface
  - Metaphors
  - A mental model
  - The navigation rules for the model
  - Look
  - Feel
- Two key questions for user interface design
  - How should the user interact with the computer system
  - How should information from the computer systems be presented to the user

6

UTSA CS3773

## User interactions

- User interaction means issuing commands and associated data to the computer system
  - Direct manipulation
  - Menu selection
  - Form fill-in
  - Command language
  - Natural language

7

UTSA CS3773

## Interface Design Process

- Interface design is an iterative process
- Interface design activities
  - User analysis - developing an understanding of the tasks that users do, their working environment, etc,
  - System prototyping - exposing the prototype systems to user to guide the evolution of the interface
  - Interface evaluation - having a more formalized evaluation activity where you collect information about the users' actual experience with the interface

8

UTSA CS3773

## Design Patterns

- We may design a series of applications that have similar functionality - want to take advantage of the commonality among systems rather than develop each "from scratch"
- A design pattern names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable design
- A design pattern addresses
  - Aggregation of classes
  - Relationships among classes
  - Communication mechanisms among classes

9

UTSA CS3773

## Design Pattern Template

- Design patterns may be described using a template
  - Pattern name
  - Intent
  - Motivation
  - Applicability
  - Structure
  - Participants
  - Collaborations

10

UTSA CS3773

## Design Classes

- Requirements classes describes some elements of the problem domain
- Design classes refines the analysis classes by providing design details and create a new set of design classes to support the business solutions
  - User interface classes
  - Business domain classes
  - Process design
  - Persistent classes
  - System classes

11

UTSA CS3773

## Software Design Concepts

- Fundamental software design concepts provide the necessary framework for getting a software right besides getting it to work
  - Modularity
    - Abstraction
    - Information hiding
  - Component independence
  - Exception identification and handling
  - Fault prevention and fault tolerance

12

UTSA CS3773

## Modularity and Abstraction

- When we consider modular solutions to any problems, many levels of abstraction can be posed
  - At the highest level of abstraction, a solution is stated in broad terms of problem domain
  - At the lower levels of abstraction, a more detailed description of the solution is provided
- Components at one level refine those in the level above
- Different levels of abstraction provide different views
- Modularity hides details and facilitates involvement
  - Each component hides a design decision from the others

13

UTSA CS3773

## Component Independent

- We strive in most designs to make the components independent of one another
- We measure the degree of component independence using two concepts
  - Low coupling
  - High cohesion

14

UTSA CS3773

## Coupling and Cohesion

- Coupling
  - Two components are highly coupled when there is a great deal of dependence between them
  - Two components are loosely coupled when they have some dependence, but the interconnections among them are weak
  - Two components are uncoupled when they have no interconnections at all
- Cohesion
  - A component is cohesive if the internal parts of the component are related to each other and to its overall purpose

15

UTSA CS3773

## Exception Handling

- Defensive design anticipates situations that might lead to problems
  - Failure to provide a service
  - Providing the wrong service or data
  - Corrupting data
- Exception handling should be embedded in design
  - Retry
  - Correct
  - Report

16

UTSA CS3773

## Characteristics of Good Software Design

- High-quality designs should have characteristics that lead to quality products
  - Ease of understanding
  - Ease of implementation
  - Ease of testing
  - Ease of modification
  - Correct translation from the requirements specification

17

UTSA CS3773

## Criteria for Good Software Design

- Guidelines that are established for evaluating good design
  - Architecture shows the logical organization of a software
    - Is created using recognizable architectural styles or patterns
    - Is composed of loose-coupled components
    - Can be implemented in an evolutionary fashion
  - Distinct representations of data, control, structure, interfaces, and components
    - Data structures are appropriate for the classes to be implemented
    - Components exhibit independent functional characteristics
    - Interfaces reduce the complexity of connections
  - Design using a repeatable method and effective notation

18

UTSA CS3773

## Software Design Evaluation and Validation

- We check a design in two different ways
  - Validation: the design satisfies all requirements specified by the customer
  - Verification: the characteristics (quality) of a good design are incorporated
- We use some techniques for helping us to perform verification and validation
  - Mathematical validation
  - Measuring design quality
  - Design reviews

19

UTSA CS3773

## Mathematical Validation of Software Design

- Break the system into processes
- Formally show that the design is correct
  - Each system process correctly transforms the input to its expected output
  - The assertions of the each process are preserved
  - The process terminates without failure

20

UTSA CS3773

## Measuring Software Design Quality

- We check a design using a set of measures
  - Coupling
  - Cohesion
  - Complexity
    - Components
    - relationships
- These measures are general

21

UTSA CS3773

## Design reviews

- We check a design with customers, analysts, prospective users, system designers, and etc. before development continues
  - Preliminary design review
    - Moderator leading the discussions
    - Secretary recording the issues
- We check a design with developers
  - Critical design review
  - Program design review

22

UTSA CS3773

## Reading Assignments

- Sommerville's Book
  - Chapter 14, "Object-oriented design"
  - Chapter 16, "User interface design"

23

UTSA CS3773