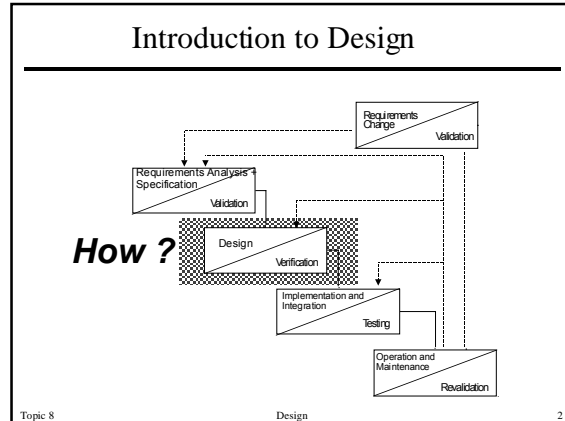


**ICS 121 Topic 8:
Design**

 Introduction to Design
 Review of Architectural Design
 Modules
 Structured Design
 Integration Testing



Goals and Objectives

- Develop a coherent representation of a software system that will satisfy the requirements
- Identify inadequacies in the requirements
- Develop review plan that demonstrates coverage of the requirements
 - yields confidence in design
- Develop test plan that covers design
 - yields confidence in both design and implementation

Topic 8 Design 3

Relationship to other lifecycle phases

- Requirements
 - Specifies the “what” not the “how”
 - Provides conceptual boundaries
 - keeps design focused
- Implementation
 - Design stops and coding begins when design specifications are sufficient for coding assignments
 - each assignment, theoretically, can be given to a programmer unaware of the overall system architecture

Topic 8 Design 4

Basic Design Process

- The design process develops several models of the software system at different levels of abstraction
 - Starting point is an informal “boxes and arrows” design
 - Add information to make it more consistent and complete
 - Provide feedback to earlier designs for improvement

Topic 8 Design 5

Design Activities

- Architectural design
 - Subsystem identification
 - services and constraints are specified
 - Module design
 - modular decomposition is performed; relationships specified
- Detailed design
 - Interface design
 - module interfaces are negotiated, designed and documented
 - Data structure and algorithm design
 - module details (data structures and algorithms) to provide system services are specified

Topic 8 Design 6

Software Architecture

- Components
 - The elements out of which the system is built
 - Examples: filters, databases, objects, ADTs
- Connectors
 - The interaction or communication mechanisms
 - The glue that combines the components
 - Examples: procedure calls, pipes, event broadcast, messages, secure protocols
- Constraints
 - Limitations on the composition of components and connectors

Topic 8 Design 13

Architectural Style

- Example architectural styles
 - Batch sequential
 - Pipe and filter
 - Main program and subroutines
 - Blackboard
 - Interpreter
 - Client-server
 - Communicating processes
 - Event systems
 - Object-oriented
 - Layered Systems

Families of systems defined by patterns of composition

Topic 8 Design 14

Architectural Design: System Structuring

- Model of the system structure and decomposition
 - how subsystems share data
 - how they are distributed
 - how they interface with each other
- Three standard models
 - **Repository model:** how subsystems exchange and share information
 - E.g., all shared data is held in a central database or each sub-system maintains its own database
 - **Distribution model:** how data and processing is distributed across a range of processors
 - E.g., Client-server or peer-to-peer processes
 - **Abstract machine model:** the interfacing of subsystems as abstract machines each of which provides a set of services to others
 - E.g., each subsystem defines an abstract machine

Topic 8 Design 15

Architectural Design: Control Modeling

- Control of subsystems so that services are delivered to the right place at the right time
- Two general approaches
 - Centralized control
 - One subsystem has overall responsibility for control and starts/stops other subsystems
 - call-return model (sequential)
 - manager model (concurrent)
 - Event-based control
 - each subsystem responds to externally generated events (from other subsystems or the environment)
 - broadcast model
 - interrupt-driven model

Topic 8 Design 16

Architectural Design: Modular decomposition

- After decomposition of the system into subsystems, subsystems must be decomposed into modules
 - There is no rigid distinction between system decomposition and modular decomposition
- Two important approaches for decomposing subsystems into modules:
 - Data-flow (structured design)
 - system is decomposed into functional modules which accept input data and transform it to output data
 - process-based decomposition
 - achieves mostly procedural abstractions
 - Object-oriented (object-oriented analysis and design)
 - system is decomposed into a set of communicating objects
 - object-based decomposition
 - achieves both procedural + data abstractions

Topic 8 Design 17


Architectural Design: Hierarchy

- Hierarchies support modular decomposition
 - *Uses relation:* a uses b only if the correct functioning of a depends on the existence of a correct implementation of b
 - modular decomposition can be specified by *uses*, where
 - Level 0 is the set of all programs that use no other program
 - Level i ($i > 0$) is the set of all programs that use at least one program on level $i-1$ and no program at level $\geq i$.
 - Note: the *uses* relation does not always provide a hierarchy
 - *Is-composed-of relation:* a is-composed-of b if b is a component of a and encapsulated within a
 - modular decomposition can be specified by *is-composed-of*, where
 - non-terminals are virtual code
 - terminals are the only units represented by code
 - Then, the *uses* relation is specified over the set of terminals only
 - Note: the *is-composed-of* relation is acyclic

Topic 8 Design 18

Types of Coupling

- content
 - one module directly references content of another
- common
 - both modules have access to same global data
- control
 - one module passes an element of control to another
- stamp
 - one module passes a data structure to another; which only uses part of the passed information
- data
 - one module passes only homogeneous data items



(Bad)

(Good)

Topic 8
Design
25

Some examples of cohesion

- Logical cohesion
 - Input/Output libraries
 - Math libraries
- Temporal cohesion
 - Program initialization
- Communicational cohesion
 - “calculate data and write it to disk”
 - Closely related: sequential cohesion
 - the output of one element is the input to another

Topic 8
Design
26

Some examples of coupling

- Control coupling
 - One module passes control flags (parameters or global variables) that control the sequence of processing steps in another module
 - Stamp coupling (alternative definition)
 - Similar to common coupling (modules that share global data) except that globals are shared selectively among routines that require the data
 - Ada packages support stamp coupling since variables defined in a package specification are shared between all modules which use the package.

Topic 8
Design
27

Structured Design

- System is completely specified by the functions that is to perform
 - Top-down, iterative refinement of functionality
 - break the [system] function into subfunctions
 - determine hierarchy and data interaction
 - Function refinement guides data refinement
 - Hierarchical organization is a tree with one module per subfunction
- Pros and Cons
 - modules are highly functional
 - best suited when state information is not pervasive
 - data decisions must be made earlier
 - changes in data ripple through entire structure
 - little chance for reusability

Topic 8
Design
28

Structured Design Process

- Identify flow of data and incorporate detail and structure iteratively
 - given specification loop
 - identify data flow and transformations
 - nouns as data
 - verbs as transformations
 - derive data flow diagrams
 - identify “natural aggregates”
 - identify highest level input and output units
 - remaining units are central transforms
 - form level of structure chart
 - control module (coordinate)
 - » input module (afferent)
 - » central module(s) (transform)
 - » output module (efferent)
 - form structure chart
 - until implementation is immediate

Topic 8
Design
29

Data Flow Diagrams

- Depict software system by flow of data from one logical processing unit (transformation) to another
 - do not include control information
 - data flow diagram elements
 - round-cornered rectangle = transformation
 - vector = data flow
 - vector operation = data flow link
 - * (and)
 - + (or)
 - + (exclusive or)
 - arc with data flow link = bracketing to override precedence
 - and over or over exclusive or
 - rectangle = data store
 - circle = user interaction (input/output)

Topic 8
Design
30

