Learning Objectives

- Describe the steps involved in the traditional approach to designing the application architecture
- Develop a system flowchart
- Develop a structure chart using transaction analysis and transform analysis
- Write pseudocode for structured modules
- Explain how to use three-layer design with the traditional approach
Overview

- Traditional approach to designing software
  - Overview of structured models, model development process, related terminology
  - How data flow diagrams are annotated with automation boundary information
  - How analysis phase models are transformed into design models using system flowcharts, structure charts, and module pseudocode
  - Integration into other design phase activities
  - Applying approach to a three-layer architecture
The Structured Approach to Designing the Application Architecture

- Application software programs
  - Designed in conjunction with database and user interface
  - Hierarchy of modules
- Design internal logic of individual modules
- Top-down approach
  - DFDs with automation boundaries
  - System flowcharts, structure charts, pseudocode
Structured Design Models

Figure 10-1
Structured design models

Data flow diagram
Structured English
data flow definitions

Data flow diagram
with automation
system boundary

System flowchart

Structure chart

If A then
Calculate Sales Tax
Calculate Total Amount
End If

Pseudocode
The Automation System Boundary

- Partitions data flow diagram processes into manual processes and automated systems
- Processes can be inside or outside boundary
- Data flows can be inside and outside of boundary
  - Data flows that cross system boundary represent inputs and outputs of system
  - Data flows that cross boundaries between programs represent program-to-program communication
DFD with Automation System Boundary (Figure 10-2)
The System Flowchart

- Representation of various computer programs, files, databases, and associated manual processes that make up complete system
- Frequently constructed during analysis activities
- Graphically describes organization of subsystems into automated and manual components
- Can show type of transaction processing system
  - Batch
  - Real-time
Common System Flowchart Symbols

- Process or program
- Input or output screen display
- File or database
- File or database
- Document or report
- Manual operation
- File on magnetic tape
- Connection between components. An arrow generally indicates a flow from one component to another.
- Communication link
Sample System Flowchart for Payroll System (Figure 10-4)
System Flowchart for RMO (Figure 10-5)
The Structure Chart

- Describes functions and subfunctions of each part of system
- Shows relationships between modules of a computer program
- Simple and direct organization
  - Each module performs a specific function
  - Each layer in a program performs specific activities
- Chart is tree-like with root module and branches
A Simple Structure Chart for the Calculate Pay Amounts Module

Figure 10-6
A simple structure chart for the Calculate pay amounts module
Structure Chart Symbols

Figure 10-7
Structure chart symbols

(a) Module
(b) Common subroutine module
(c) Boss module
(d) Boss module
(e) Called module
(f) Embedded module
(g) Boss module with iteration on called modules
(h) Boss module with a condition call
Structure Chart for Entire Payroll Program

Figure 10-8
A structure chart for the entire Payroll program

Developing a Structure Chart

- **Transaction analysis**
  - Uses system flow chart and event table inputs
  - Upper-level modules developed first
  - Identifies each transaction supported by program

- **Transform analysis**
  - Uses DFD fragments for inputs
  - Computer program “transforms” inputs into outputs
  - Charts have input, calculate, and output subtrees
Event-partitioned DFD for the Order-Entry Subsystem (Figure 10-9)
High-Level Structure Chart for the Order-Entry Subsystem After Transaction Analysis

Figure 10-10
High-level structure chart for the Customer order program
Steps to Create a Structure Chart from a DFD Fragment

◆ Determine primary information flow
  ● Main stream of data transformed from some input form to output form

◆ Find process that represents most fundamental change from input to output

◆ Redraw DFD with inputs to left and outputs to right – central transform process goes in middle

◆ Generate first draft of structure chart based on redrawn data flow
The *Create New Order* DFD Fragment

**Figure 10-11**

*The Create new order DFD fragment*
Decomposed DFD for *Create New Order*

**Figure 10-12**
Exploded view of the *Create new order* DFD
Rearranged *Create New Order* DFD

**Figure 10-13**
Rearranged view of the *Create new order* DFD

![Diagram of the rearranged Create New Order DFD](image)

- **Process**
  - 2.0: Create new order

- **Input**
  - 2.1: Record customer information
    - Afferent data flow

- **Output**
  - 2.3: Process order transaction
    - Efferent data flow

- **Central transform**
  - Customer
  - Order
  - Order transaction
  - Order item
  - Product item
  - Inventory item

- **2.4: Produce confirmation**
First Draft of the Structure Chart for *Create New Order* (Figure 10-14)
Steps to Create a Structure Chart from a DFD Fragment (continued)

- Add other modules
  - Get input data via user-interface screens
  - Read from and write to data storage
  - Write output data or reports

- Add logic from structured English or decision tables

- Make final refinements to structure chart based on quality control concepts
The Structure Chart for the *Create New Order* Program (Figure 10-15)
Combination of Structure Charts: Transaction and Transform Analysis
(Figure 10-16)
Evaluating the Quality of a Structure Chart

- **Module coupling**
  - Measure of how module is connected to other modules in program
  - Goal is to be loosely coupled

- **Module cohesion**
  - Measure of internal strength of module
  - Module performs one defined task
  - Goal is to be highly cohesive
Examples of Module Cohesion

(a) Poor cohesion

(b) Good cohesion
Module Algorithm Design—Pseudocode

- Describes internal logic of software modules
- Variation of structured English that is closer to programming code
- Syntax should mirror development language
- Three types of control statements used in structured programming
  - **Sequence** – sequence of executable statements
  - **Decision** – if-then-else logic
  - **Iteration** – do-until or do-while
Integrating Structured Application Design with Other Design Tasks

- Structure chart must be modified or enhanced to integrate design of user interface and database
  - Are additional modules needed?
  - Does pseudocode in modules need modification?
  - Are additional data couples needed to pass data?

- Structure charts and system flowcharts must correspond to planned network architecture
  - Required protocols, capacity, and security
Three-Layer Design

- Three-layer architecture
  - View layer, business logic layer, and data layer

- Structure charts and system flowcharts describe design decisions and software structuring

- Employs multiple programs for user interface, business logic, and data access modules

- Modules in different layers communicate over real-time links using well-defined protocols
System Flowchart Showing Three-Layer Architecture for Customer Order

Figure 10-19
A system flowchart showing three-layer architecture for the Customer order program
Structure Chart Showing Three-Layer Architecture for Create New Order (Figure 10-20)
Summary

- For traditional structured approach to systems design, primary input is data flow diagram
  - DFD is enhanced by adding system boundary
  - Designer describes processes within each DFD boundary using one or more structure charts
- Structure charts developed using
  - Transaction analysis – multiple transaction types
  - Transform analysis – single transaction from input to output
Summary (continued)

- Structure charts may be based on three-layer architecture
  - Modules will be clearly identified by layer
  - Structure chart may be decomposed if layers execute on multiple systems
- Structured design may also include
  - System flowcharts to show data movement
  - Module pseudocode to describe internal logic of structure chart module