Structured Paradigm Design Phase

• Input to design process is the specification document
  – Description of what the product must do
• The output from design process is the design document
  – Description of how the product is to achieve its goal

Structured Paradigm Design Phase

• Software design phase consists of three activities
  – Architectural design
    • General, Logic, or High-Level design
      – Analyze specification
      – Module structure is produced
  – Detailed design
    • Modular, Physical, or Low-Level design
      – Each module is designed in detail
  – Design testing
Structured Paradigm Design Phase

• Two basic ways of designing products in structured paradigm:
  – Action-Oriented design
  – Data-Oriented design

Structured Paradigm Design Phase

• Action-Oriented design
  • Objective is to design modules with high cohesion
  – Weaknesses
    • Concentration on action
    • Data is secondary
Structured Paradigm Design Phase

• Data-Oriented design
  • Structure of the data is determined
  • Procedures are designed to conform to the structure of data
  – Weaknesses
    • Concentration on data
    • Action is secondary

Solution To Structured Paradigm

• The solutions to these problems are
  – Object-Oriented technique
    • Equal weight to both
      – Data
      – Action
Action-Oriented Design

• Decomposing a product into modules with
  – High cohesion
  – Low coupling
• Two practical techniques to achieve this goal
  – Data flow analysis
  – Transaction analysis

Data Flow Analysis

• Technique for achieving modules with high cohesion
• Once the Data Flow Diagram is completed, the designer has precise and complete info regarding input to and output from the product
• Modules are decomposed stepwise until each module performs a single task
Data Flow Analysis

• DFA is a way to achieve high cohesion not low coupling
• In order to achieve low coupling, minor modifications to the design are needed

Transaction Analysis

• Transaction is an operation
  – Process request
  – Print a list of today’s orders
  – Example
    • Software controlling an Automated Teller Machine
Transaction Analysis

• Design strategy is to brake down the product into two pieces
  – The analyzer
    • Determines the transaction type
    • Passes the information to the dispatcher
  – The Dispatcher
    • Performs the transaction
Data-Oriented Design

• Basic idea is to:
  – Create the data structure
  – Design the product around the data structure
  – Each procedure is given the data structure as input
    on which it will operate
• After Object-Oriented paradigm it lost popularity

Object-Oriented Design

• The goal is to design the product in terms of Objects
  – Instance of classes
• OOD can be used for products utilizing
  – C
  – FORTRAN
  – COBAL
• It is not an easy task but possible
Object-Oriented Design

- OOD consist of four steps
  - Construct interaction diagrams for each scenario
  - Construct the detailed class diagram
  - Design the product in terms of client of Objects
  - Proceed to the detailed design

OOD-First Step

- UML supports two types of diagrams
  - Sequence diagram
    - Emphasizes the explicit sequence of messages
    - Important when the order of events and messages are critical
  - Collaboration diagram
    - Emphasizes the relationship between objects
    - Powerful tool in understanding the structure of product
  - Both diagrams show the same thing in different way
OOD-First Step

1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 7.
2. The floor button informs the elevator controller that the floor button has been pressed.
3. The elevator controller sends a message to the Up floor button to turn itself on.
4. The elevator controller sends a series of messages to the elevator to move itself up to floor 7. The elevator contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 7.
5. The elevator controller sends a message to the Up floor button to turn itself off.
6. The elevator controller sends a message to the elevator doors to open themselves.
7. The elevator controller starts the timer.
8. User A enters the elevator.
10. The elevator controller sends a message to the elevator button for floor 7 to turn itself on.
11. The elevator controller sends a message to the elevator doors to close themselves after a timeout.
12. The elevator controller sends a series of messages to the elevator to move itself up to floor 7.
13. The elevator controller sends a message to the elevator button for floor 7 to turn itself off.
14. The elevator controller sends a message to the elevator doors to open themselves to allow User A to exit from the elevator.
15. The elevator controller starts the timer.
16. The elevator controller controls the elevator doors to close themselves after a timeout.
17. The elevator controller sends a series of messages to the elevator to move itself up to floor 9 with User B.
OOD-First Step

OOD-Second Step

• Methods are inserted into class diagram (from OOA)
  – Examine all interaction diagrams
    • Easy to do
    – Figure out which action should be associated with each class
      • Difficult to do
OOD-Second Step

• An action can be assigned to
  – Class
    • Program unit that accepts a message
  – Client
    • Program unit that sends a message

OOD-Second Step

• Ways to decide how to assign actions
  – Based on information hiding
    • State variable and the actions performed should be assigned to that class
  – Based on popularity
    • If an action is being invoked by many clients of an object, the action should be assigned to that object
  – Based on responsibility
    • The client should not know how an action is being performed
OOD-Second Step

- Interaction diagram is to show client of each object

OOD-Third Step

- Interaction diagram is to show client of each object
OOD-Fourth Step

• Detailed design is developed for all classes
  – Pseudocode

Real-Time Design Techniques

• Real-Time software is characterized by hard time limits
• If the time constraint is not met the data is lost
• Real-Time systems are implemented on a distributed system
• Examples
  – Computer controlled nuclear power plant
  – Software controlling a fighter aircraft
Testing During The Design Phase

• The goal of testing is
  – Verify all items in specification have been implemented accurately and completely
  – Verify the design
    • logic does not contain faults
    • Interfaces have been defined correctly
• Important to correct all faults before coding

Testing During The Design Phase

• Design faults can be detected by
  – Design inspection
  – Design walkthroughs
Metrics For The Design Phase

• Keeping track of
  – Number of modules
  – Module cohesion and coupling are measured for quality of design
  – Fault statistics
  – Number and type of faults discovered during inspection or walkthrough

Challenges Of The Design Phase

• Design team can go wrong in two ways
  – By doing too much
    • Designers who enjoy coding, write modules in C++ or JAVA instead of pseudocode
  – By doing too little
    • Leave the majority of the detail design to programmer

• Insuring that all interfaces are correct is the primary reason for detail design