CHAPTER 6

THE TRADITIONAL APPROACH TO REQUIREMENTS
Learning Objectives

◆ Explain how the traditional approach and the object-oriented approach differ when modeling the details of a use case

◆ List the components of a traditional system and the symbols representing them on a data flow diagram

◆ Describe how data flow diagrams can show the system at various levels of abstraction
Learning Objectives (continued)

- Develop data flow diagrams, data element definitions, data store definitions, and process descriptions

- Read and interpret Information Engineering models that can be incorporated within traditional structured analysis

- Develop tables to show the distribution of processing and data access across system locations
Overview

- What the system does and what event occurs – activities and interactions (use case)
- Traditional structured approach to representing activities and interactions
- Diagrams and other models of the traditional approach
- RMO customer support system example shows how each model is related
- How traditional and IE approaches and models can be used together to describe system
Traditional versus Object-Oriented Approaches

**Figure 6-1**

Traditional versus OO approaches

**Traditional Approach**
- System is a collection of processes
- Processes interact with data entities
- Processes accept inputs and produce outputs

**OO Approach**
- System is a collection of interacting objects
- Objects interact with people and each other
- Objects send and respond to messages
Traditional Approach in this Chapter

Figure 6-2
Requirements models for the traditional and OO approaches

Traditional Approach
- Context diagram
- Data flow definitions
- Other traditional models

Class diagram
- Use case diagrams
- System sequence diagrams
- State machine diagrams

Object-Oriented Approach
- DFD fragments
- Process descriptions
- Use case descriptions
- Activity diagrams
Data Flow Diagrams (DFDs)

- Graphical system model that shows all main requirements for an IS in one diagram
  - Inputs/outputs
  - Processes
  - Data storage

- Easy to read and understand with minimal training
Data Flow Diagram Symbols
(Figure 6-3)

- **Process**: Step-by-step instructions are followed that transform inputs into outputs (a computer or person or both doing the work).
- **Data flow**: Data flowing from place to place, such as an input or output to a process.
- **External agent**: The source or destination of data outside the system.
- **Data store**: Data at rest, being stored for later use. Usually corresponds to a data entity on an entity-relationship diagram.
- **Real-time link**: Communication back and forth between an external agent and a process as the process is executing (e.g., credit card verification).
DFD Fragment Showing Use Case *Look up item availability* from the RMO (Figure 6-4)
DFD Integrates Event Table and ERD (Figure 6-5)
DFD and Levels of Abstraction

- Data flow diagrams (DFDs) are decomposed into additional diagrams to provide multiple levels of detail.

- Higher-level diagrams provide general views of system.

- Lower-level diagrams provide detailed views of system.

- Differing views are called levels of abstraction.
Layers of DFD Abstraction for Course Registration System (Figure 6-6)
Context Diagrams

- DFD that summarizes all processing activity for the system or subsystem
- Highest level (most abstract) view of system
- Shows system boundaries
- System scope is represented by a single process, external agents, and all data flows into and out of the system
DFD Fragments

- Created for each use case in the event table
- Represent system response to one event within a single process symbol
- Self-contained models
- Focus attention on single part of system
- Show only data stores required in the use case
Three Separate DFD Fragments for Course Registration System

**Figure 6-7**

DFD fragments for the course registration system

- **1**: Academic department → Schedule course → Offered course
- **2**: Student → Enroll student → Course enrollment
- **3**: Faculty member → Produce class list → Offered course
Event-Partitioned System Model

- DFD to model system requirements using single process for each use case/activity in system or subsystem
- Combines all DFD fragments together to show decomposition of the context-level diagram
- Sometimes called “diagram 0”
- Used primarily as a presentation tool
- Decomposed into more detailed DFD fragments
Combining DFD Fragments to Create Event-Partitioned System Model (Figure 6-8)
Context Diagram for RMO Customer Support System (Figure 6-9)
### RMO Subsystems and Use Cases/Activities from Event Table (Figure 6-10)

<table>
<thead>
<tr>
<th>Order-entry subsystem</th>
<th>Customer maintenance subsystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up item availability</td>
<td>Provide catalog information</td>
</tr>
<tr>
<td>Create new order</td>
<td>Produce prospective customer activity reports</td>
</tr>
<tr>
<td>Update order</td>
<td>Update customer account</td>
</tr>
<tr>
<td>Produce order summary reports</td>
<td>Distribute promotional package</td>
</tr>
<tr>
<td>Produce transaction summary reports</td>
<td>Create customer charge adjustment</td>
</tr>
<tr>
<td></td>
<td>Produce customer adjustment reports</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order fulfillment subsystem</th>
<th>Catalog maintenance subsystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up order status</td>
<td>Update catalog</td>
</tr>
<tr>
<td>Record order fulfillment</td>
<td>Create special product promotion</td>
</tr>
<tr>
<td>Record back order</td>
<td>Create new catalog</td>
</tr>
<tr>
<td>Create order return</td>
<td>Produce catalog activity reports</td>
</tr>
<tr>
<td>Produce fulfillment summary report</td>
<td></td>
</tr>
</tbody>
</table>
Context Diagram for RMO Order-Entry Subsystem (Figure 6-11)
Five Separate DFD Fragments for RMO Order-Entry Subsystem (Figure 6-12)
Decomposing DFD Fragments

- Most DFD fragments can be further described using structured English.

- Sometimes DFD fragments need to be diagrammed in more detail.

- Decomposed into subprocesses in a detailed DFD.

- DFD numbering scheme.
  - Hierarchical decomposition.
    - DFD Fragment 2 is decomposed into Diagram 2.
    - Diagram 2 has processes 2.1, 2.2, 2.3, 2.4.
Detailed DFD for *Create new order* DFD Fragment (Figure 6-14)
Physical and Logical DFDs

◆ Logical model
  ● Assumes implementation in perfect technology
  ● Does not tell how system is implemented

◆ Physical model
  ● Describes assumptions about implementation technology
  ● Developed in last stages of analysis or in early design
Physical DFD for Scheduling Courses (Figure 6-15)
Evaluating DFD Quality

- Readable
- Internally consistent and balanced
- Accurately represents system requirements
- Reduces information overload – rule of 7 +/- 2
  - Single DFD should not have more than 7 +/- 2 processes
  - No more than 7 +/- 2 data flows should enter or leave a process or data store in a single DFD
- Minimizes required number of interfaces
Data Flow Consistency Problems

- Differences in data flow content between a process and its process decomposition
- Data outflows without corresponding inflows
- Data inflows without corresponding outflows
- Results in unbalanced DFDs
Consistency Rules

◆ All data that flows into a process must
  ● Flow out of the process, or
  ● Be used to generate data that flows out of the process

◆ All data that flows out of a process must
  ● Have flowed into the process, or
  ● Have been generated from data that flowed into the process
Unnecessary Data Input: Black Hole

**Figure 6-16**
A process with unnecessary data input—a black hole
Process with Impossible Data Output: A Miracle (Figure 6-17)
Process with Unnecessary Data Input
(Figure 6-18)
Process with Impossible Data Output
(Figure 6-19)

Process description:
If A>5 Then
  Y=100
Else
  Y=250
Endif
Documentation of DFD Components

- Lowest-level processes need to be described in detail

- Data flow contents need to be described

- Data stores need to be described in terms of data elements

- Each data element needs to be described

- Various options for process definition exist
Structured English

- Method of writing process specifications

- Combines structured programming techniques with narrative English

- Well-suited for lengthy sequential processes or simple control logic (single loop or if-then-else)

- Ill-suited for complex decision logic or few (or no) sequential processing steps
Structured English Example (Figure 6-20)

Process Ballots Procedure

Collect all ballots
Place all ballots in a stack
Set Yes count and No count to zero
Repeat for each ballot in the stack
  If Yes is checked then
    Add one to Yes count
  Else
    Add one to No count
  Endif
  Place ballot on counted ballot stack
Endrepeat
If Yes count is greater than No count then
  Declare Yes the winner
Else
  Declare No the winner
Endif
Store the counted ballot stack in a safe place
End Process Ballots Procedure
Process 2.1 - Record Customer Information

Ask if customer has an account (or has made a previous order)
If customer has an account then
   Ask for identification information
   Query database with identifying information
   Copy query response data to Order details
Else
   Create an empty Customer record in the database
   Ask customer for Customer attributes
   Update empty Customer record with Customer attributes
Endif
Ask customer for order information for first item
While more order items Do
   Update Order details with order information
Endwhile
Decision Tables and Decision Trees

- Can summarize complex decision logic better than structured English
- Incorporate logic into the table or tree structure to make descriptions more readable

<table>
<thead>
<tr>
<th>YTD purchases &gt; $250</th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Items (N)</td>
<td>N ≤ 3</td>
<td>N ≥ 4</td>
</tr>
<tr>
<td>Delivery Day</td>
<td>Next</td>
<td>2nd</td>
</tr>
<tr>
<td>Shipping Charge ($)</td>
<td>25</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 6-23
A decision table for calculating shipping charges
Decision Tree for Calculating Shipping Charges (Figure 6-24)
Data Flow Definitions

- Textual description of data flow’s content and internal structure
- Often coincide with attributes of data entities included in ERD plus computed values
- Algebraic notion describes data elements on data flow plus data structure

New-Order = Customer-Name + Customer-Address + Credit-Card-Information + \sum_{i=1}^{n} \{ Item-Number + Quantity \}
Data Flow Definition for RMO Products and Items Control Break Report (Figure 6-29)

```
products-and-items-report =

"\{ product-id + product-name + season + category +
 supplier + unit-price + special + special-price +
 discontinued + description +
 \{ size + color + style + units-in-stock +
   recorder-level + units-on-order
 \}
\}
```
Data Element Definitions

- Data type description
  - String, integer, floating point, Boolean
  - Sometimes very specific written description

- Length of element

- Maximum and minimum values

- **Data dictionary** – repository for definitions of data flows, data stores, and data elements
Data Element Definition Examples
(Figure 6-30)

units-in-stock =
   a positive integer

supplier =
   a four digit numeric code

unit-price =
   a positive real number accurate to two decimal places,
   always in U.S. dollars

description =
   a text field containing a maximum of 50 printable characters

special =
   a coded field with one of the following values
   0: item is not “on special”
   1: item is “on special”
Components of a Traditional Analysis Model
(Figure 6-31)
Information Engineering Models

- Focus on strategic planning, enterprise applications, and data requirements of new system
- Share features with structured system development methodology
- Developed by James Martin in early 1980’s
- Thought to be more rigorous and complete than the structured approach
Information Engineering System Development Life Cycle Phases (Figure 6-32)
Process Decomposition and Dependency Models

- IE process models show three information types
  - Decomposition of processes into other processes
  - Dependency relationships among processes
  - Internal processing logic

- Process decomposition diagram – represents hierarchical relationship among processes at different levels of abstraction

- Process dependency model – describes ordering of processes and interaction with stored entities
Process Decomposition Diagram for RMO (Figure 6-34)
Process Dependency Diagram (Figure 6-35)
Locations and Communication Through Networks

- Logical information needed during analysis
  - Number of user locations
  - Processing and data access requirements at various locations
  - Volume and timing of processing and data access requests

- Needed to make initial design decisions such as
  - Distribution of computer systems, application software, database components, network capacity
Gathering Location Information

- Identify locations where work is to be performed
- Draw location diagram
- List functions performed by users at each location

Build activity-location matrix
- Rows are system activities from event table
- Columns are physical locations

Build activity-data (CRUD) matrix
- CRUD – create, read, update, and delete
### RMO Activity-Location Matrix (Figure 6-38)

<table>
<thead>
<tr>
<th>ACTIVITY</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Corporate offices (Park City)</td>
</tr>
<tr>
<td>Look up item availability</td>
<td>X</td>
</tr>
<tr>
<td>Create new order</td>
<td></td>
</tr>
<tr>
<td>Update order</td>
<td></td>
</tr>
<tr>
<td>Look up order status</td>
<td>X</td>
</tr>
<tr>
<td>Record order fulfillment</td>
<td></td>
</tr>
<tr>
<td>Record back order</td>
<td></td>
</tr>
<tr>
<td>Create order return</td>
<td></td>
</tr>
<tr>
<td>Provide catalog info</td>
<td></td>
</tr>
<tr>
<td>Update customer account</td>
<td>X</td>
</tr>
<tr>
<td>Distribute promotional package</td>
<td></td>
</tr>
<tr>
<td>Create customer charge adjustment</td>
<td></td>
</tr>
<tr>
<td>Update catalog</td>
<td>X</td>
</tr>
<tr>
<td>Create special product promotion</td>
<td></td>
</tr>
<tr>
<td>Create new catalog</td>
<td>X</td>
</tr>
</tbody>
</table>
### RMO Activity-Data Matrix (CRUD)
(Figure 6-39)

<table>
<thead>
<tr>
<th>ACTIVITIES</th>
<th>Catalog</th>
<th>Customer</th>
<th>Inventory item</th>
<th>Order item</th>
<th>Order transaction</th>
<th>Package</th>
<th>Product item</th>
<th>Return item</th>
<th>Shipment</th>
<th>Shlippers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up item availability</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create new order</td>
<td>CRU</td>
<td>RU</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>R</td>
<td>R</td>
<td>C</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Update order</td>
<td>RU</td>
<td>RU</td>
<td>RUD</td>
<td>RUD</td>
<td>RUD</td>
<td>R</td>
<td>R</td>
<td>CRUD</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Look up order status</td>
<td>R</td>
<td></td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Record order fulfillment</td>
<td></td>
<td></td>
<td>RU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record back order</td>
<td>RU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CRUD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create order return</td>
<td>CRU</td>
<td>RU</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td>C</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide catalog info</td>
<td>R</td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update customer account</td>
<td>CRUD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distribute promotional package</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create customer charge adjustment</td>
<td>RU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CRUD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update catalog</td>
<td>RU</td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>RU</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create special product promotion</td>
<td>R</td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create new catalog</td>
<td>C</td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>CRU</td>
<td>R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C = Creates new data, R = Reads existing data, U = Updates existing data, D = Deletes existing data
Summary

- Data flow diagrams (DFDs) are used in combination with event table and entity-relationship diagram (ERD) to model system requirements.
- DFDs model system as set of processes, data flows, external agents, and data stores.
- DFDs easy to read – graphically represent key features of system using small set of symbols.
- Many types of DFDs – context diagrams, DFD fragments, subsystem DFDs, event-partitioned DFDs, and detailed process DFDs.
Summary (continued)

- Each process, data flow, and data store requires detailed definition.
- Analyst may define processes as structured English process specifications, decision tables, decision trees, or detail process DFDs.
- Detailed process decomposition DFDs used when internal process complexity is great.
- Data flows are defined by component data elements and their internal structure (algebraic notation).
Models from IE may supplement DFDs

- Process decomposition diagram (how processes on multiple DFD levels are related)
- Process dependency diagram (emphasizes interaction with stored entities)
- Location diagram (where system is used)
- Activity-location matrix (which processes are implemented at which locations)
- Activity-data (or CRUD) matrix (where data is used)