Chapter 6: The Traditional Approach to Requirements

Systems Analysis and Design in a Changing World, 3rd Edition

Learning Objectives

- Explain how the traditional approach and the object-oriented approach differ when an event occurs
- 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)

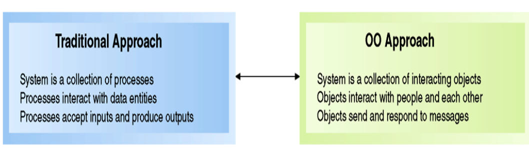
- Develop data flow diagrams, data element definitions, data store definitions, and process descriptions
- Develop tables to show the distribution of processing and data access across system locations
- Read and interpret Information Engineering models that can be incorporated within traditional structured analysis

Overview

- What the system does what an event occurs: activities and interactions
- 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

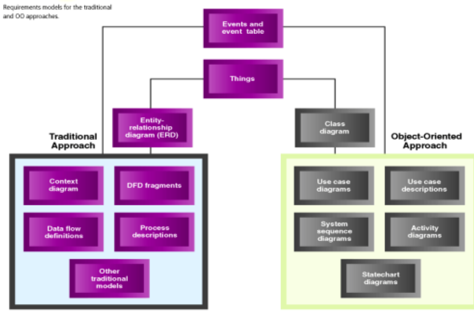
Traditional and Object-Oriented Views of Activities

FIGURE 6-1
Traditional versus OO approaches.



Requirements Models for the Traditional and OO Approaches

FIGURE 6-2
Requirements models for the traditional and OO approaches.



Data Flow Diagrams

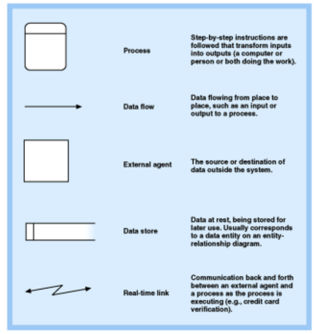
6

- 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

6

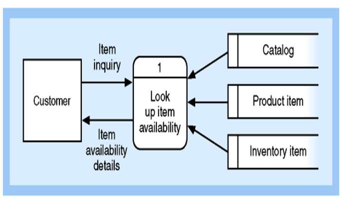
FIGURE 6-3
Data flow diagram symbols.



DFD Fragment from the RMO Case

6

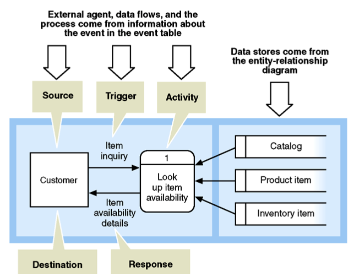
FIGURE 6-4
A DFD showing the process *Look up item availability* (a DFD fragment from the RMO case).



DFD Integrates Event Table and ERD

6

FIGURE 6-5
The DFD integrates the event table and the ERD.



DFD and Levels of Abstraction

6

- 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

6

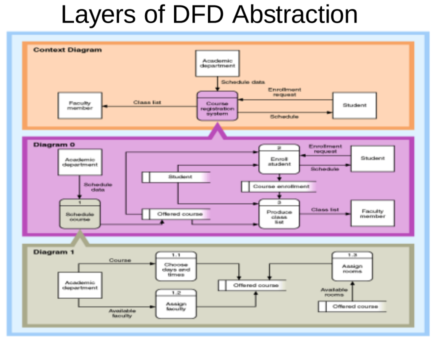


FIGURE 6-6
Layers of DFD abstraction for a course registration system.

6

Context Diagrams

- DFD that summarizes all processing activity
- 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

6

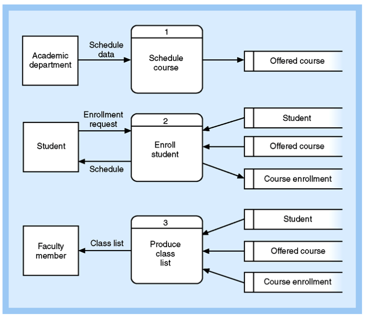
DFD Fragments

- Created for each event in the event table
- Represents system response to one event within a single process symbol
- Self contained model
- Focuses attention on single part of system
- Shows only data stores required to respond to events

6

DFD Fragments for Course Registration System

FIGURE 6-7
DFD fragments for the course registration system.



6

Event-Partitioned System Model

- DFD to model system requirements using single process for each event in system or subsystem
- Decomposition of the context level diagram
- Sometimes called diagram 0
- Used primarily as a presentation tool
- Decomposed into more detailed DFD fragments

6

Combining DFD Fragments

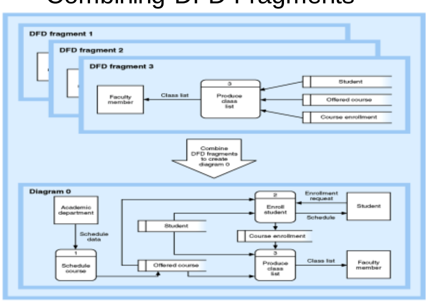


FIGURE 6-8
Combining DFD fragments to create the event-partitioned system model for the course registration system.

6

Context Diagram for RMO Customer Support System

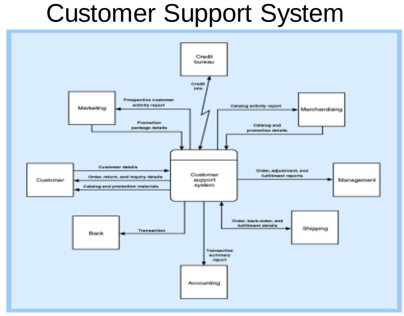


FIGURE 6-9
A context diagram for the RMO customer support system.

RMO Subsystems and Events

FIGURE 6-10
RMO subsystems and events for each subsystem.

Order-entry subsystem

- Customer wants to check item availability
- Customer places an order
- Customer changes or cancels an order
- Time to produce order summary reports
- Time to produce transaction summary reports

Order fulfillment subsystem

- Customer or management wants to check order status
- Shipping fulfills an order
- Shipping identifies a back order
- Customer returns the item (the item is defective, the customer has changed his mind, full or partial returns)
- Time to produce fulfillment summary reports

Customer maintenance subsystem

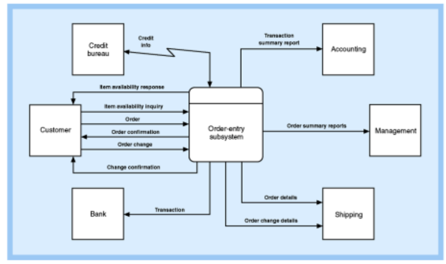
- Prospective customer requests a catalog
- Time to produce prospective customer activity reports
- Customer updates account information
- Marketing wants to send promotional materials to customers
- Management adjusts customer charges (correct errors, make concessions)
- Time to produce customer adjustment/concession reports

Catalog maintenance subsystem

- Merchandising updates the catalog (add, change, delete, change prices)
- Merchandising creates a special product promotion
- Merchandising creates a new catalog
- Time to produce catalog activity reports

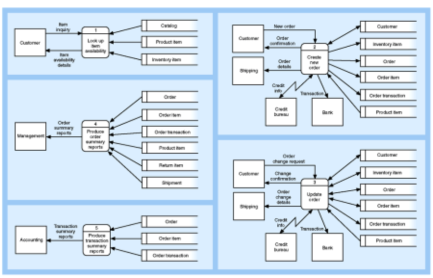
Context Diagram for RMO Order-Entry Subsystem

FIGURE 6-11
A context diagram for the RMO order-entry subsystem.



DFD Fragments for RMO Order-Entry System

FIGURE 6-12
DFD fragments for the RMO order-entry subsystem.



Decomposing DFD Fragments

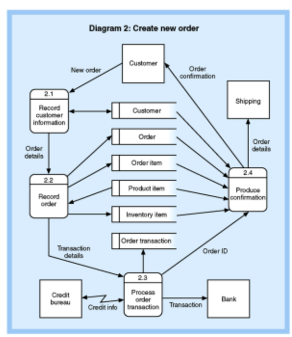
- Sometimes DFD fragments need to be explored in more detail
- Broken into subprocesses with additional detail
- DFD numbering scheme:
  - Does not equate to subprocess execution sequence
  - It is just a way for analyst to divide up work

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

Detailed Diagram for *Create New Order*

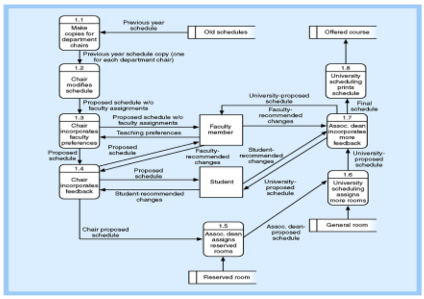
FIGURE 6-14
A detailed diagram for *Create new order* (diagram 2).



Physical DFD for scheduling courses

FIGURE 6-15
A physical DFD for scheduling courses (resist the temptation to create physical DFDs during analysis).

- Technology-specific processes
- Actor-specific process names
- Technology- or actor-specific process orders
- Redundant processes, data flows, and files



Evaluating DFD Quality

- Readable
- Internally consistent
- Accurately represents system requirements
- Reduces information overload: Rule of 7 +/- 2
  - Single DFD should have not more than 7 +/-2 processes
  - No more than 7 +/- 2 data flows should enter or leave a process or data store on a single DFD
- Minimizes required number of interfaces

Data Flow Consistency Problems

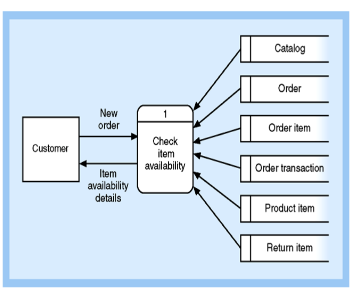
- Differences in data flow content between a process and its process decomposition
- Data outflows without corresponding inflows
- Data inflows without corresponding outflows
- Results in unbalanced DFDs

Consistency Rules

- All data that flows into a process must:
  - Flow out of the process or
  - Be used to generate data that flow 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

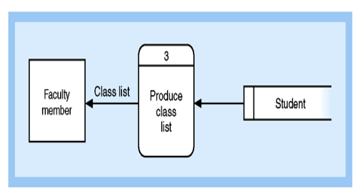
Unnecessary Data Input: Black Hole

FIGURE 6-16
A process with unnecessary data input—a black hole.



Process with Impossible Data Output: Miracle

FIGURE 6-17
A process with an impossible data output—a miracle.

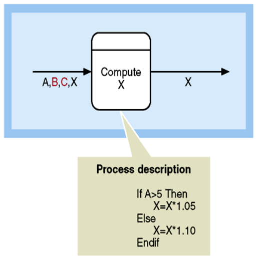


Process with Unnecessary Data Input

6

FIGURE 6-18

A process with unnecessary data input.

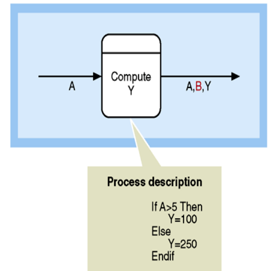


Process with Impossible Data Output

6

FIGURE 6-19

A process with an impossible data output.



Documentation of DFD Components

6

- 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

6

- Method of writing process specifications
- Combines structured programming techniques with narrative English
- Well suited to 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

6

FIGURE 6-20

A structured English example.

```
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 and Structured English Process Description

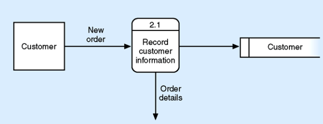
6

FIGURE 6-21

RMO process 2.1 *(Record customer information)* and its structured English process description.

```
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
```



6

Decision Tables and Decision Trees

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

FIGURE 6-23 A decision table for calculating shipping charges.

YTD purchases > $250	YES						NO					
Number of Items (N)	N≤3			N≥4			N≤3			N≥4		
Delivery Day	Next	2nd	7th	Next	2nd	7th	Next	2nd	7th	Next	2nd	7th
Shipping Charge ($)	25	10	N*1.50	N*6.00	N*2.50	Free	35	15	10	N*7.50	N*3.50	N*2.50

6

Decision Tree for Calculating Shipping Charges

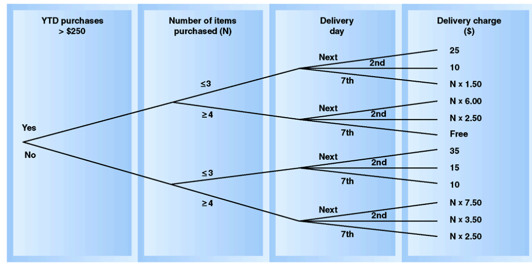


FIGURE 6-24
A decision tree for calculating shipping charges.

6

Data Flow Definitions

- Textual description of data flow's content and internal structure
- Often coincide with attributes of data entities included in ERD

FIGURE 6-26
Data flow definitions simply listing elements.

```
Customer-Name
Customer-Address
Credit-Card-Information
Item-Number
Quantity
```


6

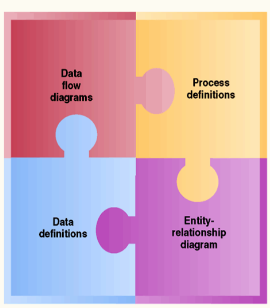
Data Element Definitions

- Data type description
  - e.g. string, integer, floating point, Boolean
  - Sometimes very specific
- Length of element
- Maximum and minimum values
- Data dictionary – repository for definitions of data flows, data stores, and data elements

6

Components of a Traditional Analysis Model

FIGURE 6-31
The components of a traditional systems analysis model.



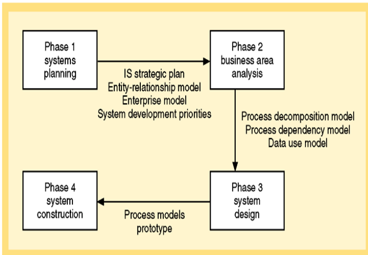
6

Information Engineering Models

- Focuses on strategic planning, enterprise size, and data requirements of new system
- Shares 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
Information Engineering systems development life cycle phases.

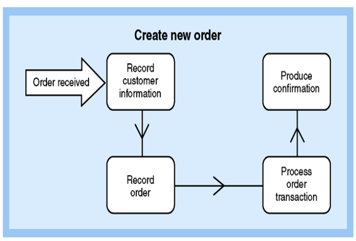


Process Decomposition and Dependency Models

- 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 Dependency Diagram

FIGURE 6-35
A process dependency diagram.



Process Dependency Diagram with Data Flows

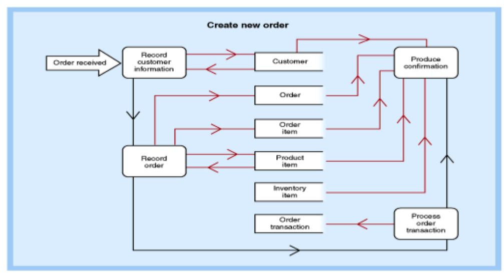


FIGURE 6-36
A process dependency diagram with data flows.

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 Location Diagram

FIGURE 6-37

The Rocky Mountain Outfitters location diagram.

RMO Activity-Location Matrix

ACTIVITY	LOCATION				
	Corporate offices (Park City)	Distribution warehouses (Salt Lake City, Albuquerque, Portland)	Mail-order (Provo)	Phone sales (Salt Lake City)	Customer direct interaction (Anticipated)
Look up item availability	X	X	X	X	X
Create new order			X	X	X
Update order				X	X
Look up order status	X	X		X	X
Record order fulfillment		X			
Record back order		X			
Create order return		X			
Provide catalog info			X	X	X
Update customer account	X		X	X	X
Distribute promotional package	X				
Create customer charge adjustment	X				
Update catalog	X				
Create special promotion	X				
Create new catalog	X				

FIGURE 6-38

Activity-location matrix for the Rocky Mountain Outfitters customer support system.

Summary

- Data flow diagrams (DFDs) 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 process decomposition DFDs

Summary (continued)

- Each process, data flow, and data store requires detailed definition
- Analyst may define processes as structured English process specification, decision table, decision tree, or process decomposition DFD
- Process decomposition DFDs used when internal process complexity is great
- Data flows defined by component data elements and their internal structure

Summary (continued)

- 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 (geographic where system used)
  - Activity-location matrix (which processes are implemented at which locations)
  - Activity-data (or CRUD) matrix (where data used)