



CS 4400 Introduction to Database Systems

Functional Data Flow Diagrams

Ling Liu

Associate Professor

College of Computing, Georgia Tech

Outline

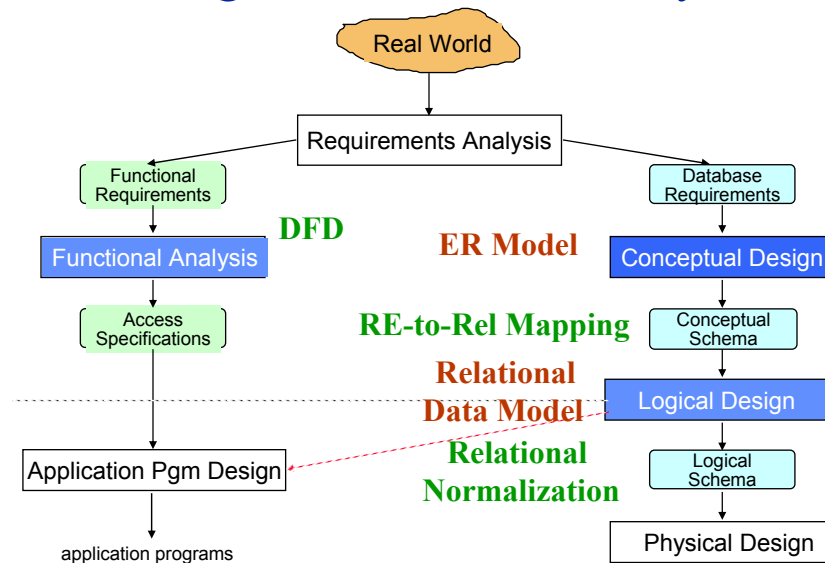
■ This week:

- Exercise of Relational Algebra
- Techniques for Data Flow Diagramming

■ Next Week

- Midterm 1 review on Tuesday
- Midterm 1 on Thursday
 - Close Book Exam or
 - Open Book Exam
 - ☞ Only materials allowed are notes and the required textbook of the course.
 - ☞ No interaction or exchange is allowed among students

DB Design in Information Systems



Representing Systems With Data Flow Diagrams

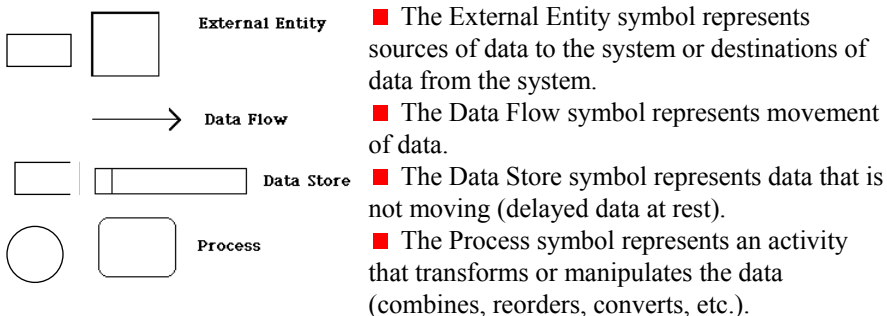
- Data flow diagrams are a network representation of a system. They are the cornerstone for structured systems analysis and design. The diagrams use four symbols to represent any system at any level of detail. The four entities that must be represented are:
 - **data flows** - movement of data in the system
 - **data stores** - data repositories for data that is not moving
 - **processes** - transforms of incoming data flow(s) to outgoing data flow(s)
 - **external entities** - sources or destinations outside the specified system boundary

DFD Purpose

- **The purpose of data flow diagrams is to provide a semantic bridge between users and systems developers. The diagrams are:**
 - graphical, eliminating thousands of words;
 - logical representations, modeling **WHAT** a system does, rather than physical models showing **HOW** it does the work;
 - hierarchical, showing systems at any level of detail; and
 - jargonless, allowing user understanding and reviewing.

DFD Description

- **Data Flow Diagrams are composed of the four basic symbols shown below.**



Any system can be represented at any level of detail by these four symbols.

External Entities

1. **are named with appropriate name.**
2. **can be duplicated, one or more times, on the diagram to avoid line crossing.**
3. **determine the system boundary. They are external to the system being studied. They are often beyond the area of influence of the developer.**
4. **can represent another system or subsystem.**
5. **go on margins/edges of data flow diagram.**

Data Flows

1. **are represented with a line with an arrowhead on one end.**
 - **A fork in a data flow means that the same data goes to two separate destinations.**
 - **The same data coming from several locations can also be joined.**
2. **should only represent data, not control.**
3. **are ALWAYS named. Name is not to include the word "data".**
4. **are referenced by a combination of the identifiers of the constructs that the data flow connects. (14-A references a data flow from process 14 to external entity A)**

Data Flow v.s. Control Flow

■ Data flow

- A data flow moves data between processes or between processes and data stores. It represents a data value at some point within a computation and an intermediate value within a computation if the flow is internal to the diagram. This value is not changed.
- The names of input and output flows can indicate their roles in the computation or the type of the value they move.
- Data names are preferably nouns. The name of a typical piece of data, the data aspect, is written alongside the arrow.

■ Control flow

- A control flow is a signal that carries out a command or indicates that something has occurred. A control flow occurs at a discrete point in time. The arrow indicates the direction of the control flow. The name of the event is written beside the arrow.

Data Stores

1. are generic for physical files (index cards, desk drawers, magnetic disk, magnetic tape, shirt pocket, human memory, etc.)
2. are named with an appropriate name, not to include the word "file", and numbered with a number preceded with a capital letter D
3. can be duplicated, one or more times, to avoid line crossing.
4. can show two or more systems that share a data store. This is done by adding a solid stripe on the left boundary.
5. are detailed in the data dictionary or with data description diagrams.

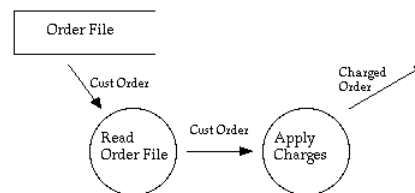
Processes

1. show data transformation or change.
 - All processes must have inputs and outputs. In some (rare) cases, data inputs or outputs will only be shown at more detailed levels of the diagrams. Each process is always "running" and ready to accept data.
2. are represented by a rounded corner rectangle (or oval or circle)
3. are named with one carefully chosen verb and an object of the verb. There is no subject.
4. have physical location shown only for existing physical systems or a physical design is being represented.
5. are numbered within the diagram as convenient. Levels of detail are shown by decimal notation.
 - For example, top level process would be Process 14, next level of detail Processes 14.1-14.4, and next level with Processes 14.3.1-14.3.6.
6. should generally move from top to bottom and left to right.

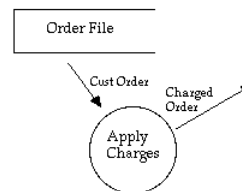
Data Flow Diagrams

- 1. Use process bubbles to show processing or transformation of data. It is not necessary to show steps where data is not transformed or processed, e.g., reading a record from a file.

NOT LIKE THIS:



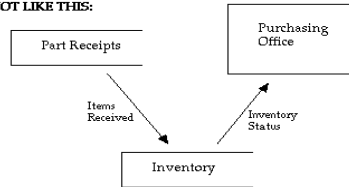
LIKE THIS:



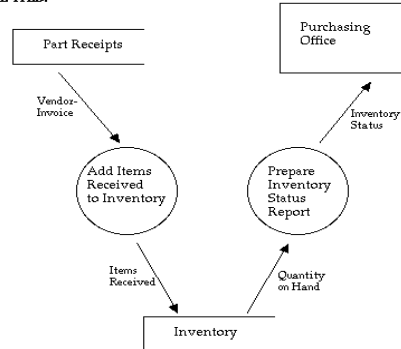
Data Flow Diagrams

- 2. Data flows must begin and/or end at a process bubble. That is, at least one end of a data flow arrow must be at a process bubble.

NOT LIKE THIS:



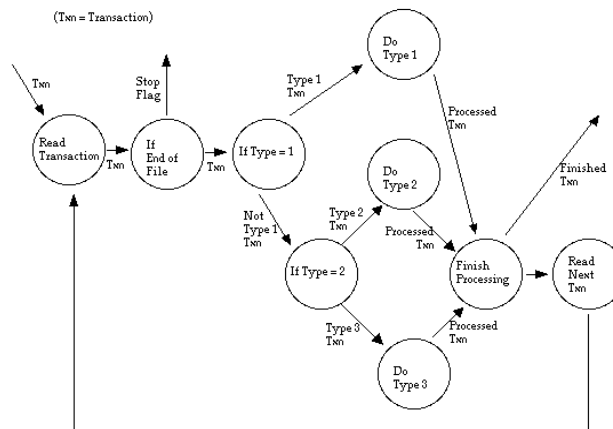
LIKE THIS:



Data Flow Diagrams

- 3. Show only data flows, not control flows (such as IF-ELSE logic).

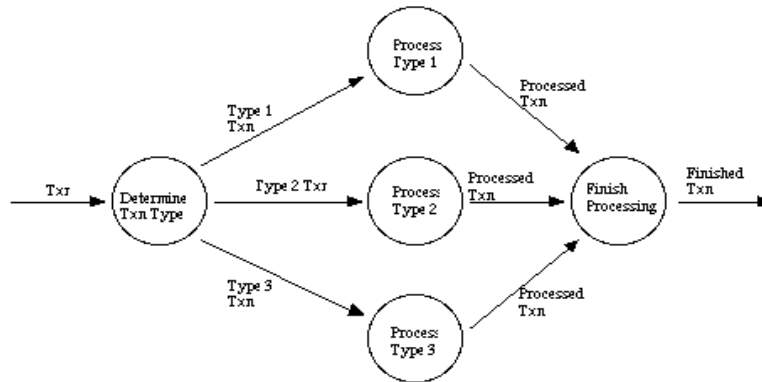
NOT LIKE THIS:



Data Flow Diagrams

- 3. Show only data flows, not control flows (such as IF-ELSE logic).

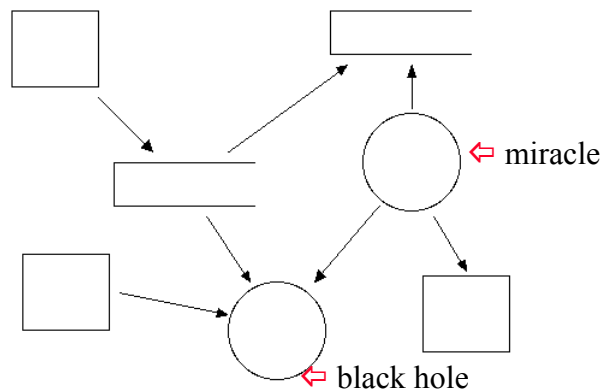
LIKE THIS:



- 4. Avoid black holes (a process bubble that has inputs but no output) and miracles (a process bubble that has outputs but not input).

Practice Exercise

- Locate the errors in the following abstract data flow diagram.



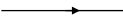
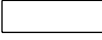



Data Flow Diagrams - Naming

- **Data Flows and Data Stores should receive names that describe the composition of the data involved. (nouns)**
- **Process bubbles should be named using strong, active verbs and objects to stress the basic data transformation or process that is occurring in that bubble.**
- **Fuzzy names for data flows and data stores, and weak or mushy verbs for processes usually indicate a lack of understanding about what is happening within the system. Thus, a periodic evaluation of the names being assigned can serve as a guide to areas where more analysis needs to be done.**

Summary: The Data Flow Model for Functional Analysis

- The data flow model support the concepts of process, dataflow, data store, and interface (external entity).
- A process represents an activity that can generate, use, manipulate, or destroy information.
- A dataflow is an exchange between processes.
- A data store is a repository of information. E.g., temporary files, look-up tables, electronic/paper forms, etc..
- An interface is an external user of the DB system, who may be the receiver of the dataflows or data stores.

Concept	Diagrammatic Representation	
Process		
Flow		
Data Store		
Interface/External Entity		

Procedure for Creating DFDs

- **The procedure for producing a data flow diagram is to:**
 1. identify and list **external entities** providing inputs/receiving outputs from system;
 2. identify and list **inputs** from/**outputs** to external entities;
 3. create a **context diagram** with system at center and external entities sending and receiving data flows;
 4. identify the **business functions** included within the system boundary;
 5. identify the **data connections** between business functions;
 6. confirm through personal contact sent data is received and vice-versa;

Procedure for Creating DFDs

- **The procedure for producing a data flow diagram is to:**
 7. trace and record **what happens to each of the data flows entering the system** (data movement, data storage, data transformation/processing)
 8. attempt to **connect any diagram segments** into a rough draft;
 9. **verify** all data flows have a **source** and **destination**;
 10. **verify** data coming out of a **data store** goes in;
 11. redraw to simplify--ponder and question result;
 12. review with "informed";
 13. explode and repeat above steps as needed.

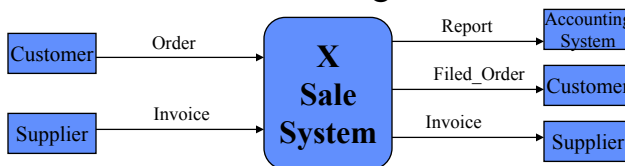
Strategies for Data Flow Design

As using ER model for conceptual DB design, there are four commonly used strategies for functional analysis using data flow diagram (DFD).

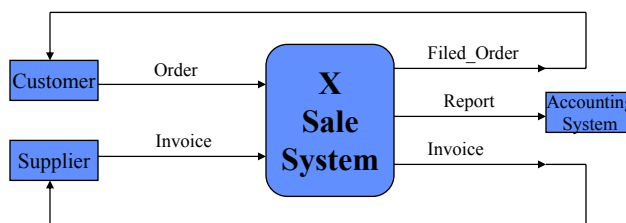
- Top-down strategy
 - Decomposition
- Bottom-up strategy
 - integration
- Mixed strategy (Inside-out strategy)

Top-Down

- Start with the content diagram



- Remove the duplicate external entities



Decomposition

- To specify what a high-level process does, break it down into smaller units in more DFDs.
- A high-level process is an entire DFD. Each high-level process is decomposed into other processes with data flows and data stores. Each decomposition is a DFD in itself.
- You can continue to break down processes until you reach a level on which further decomposition seems impossible or meaningless.
 - The data flows of the opened process are connected in the new diagram to the process related to the opened process. Vertices, and the flows and objects connected to them, are transferred with the flows that are connected to the decomposed process.

Example: Top-down Strategy

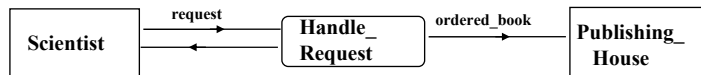
- Consider the ordering and purchasing of books by the library of a university department.



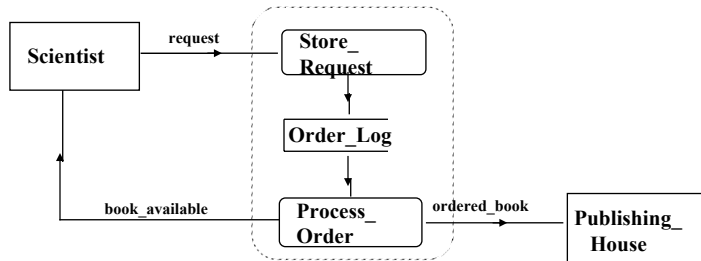
(a) Skeleton Context Diagram

- The process Handle_Request models all the ordering and purchasing activities.
- Assume:
 - T1: order processing consists of checking the feasibility of a purchase, followed by compiling an order.
 - T2: the policy of the librarian is to collect book orders and process them periodically.
 - T3: the check_book activity can be decomposed into two parallel activities: checking
 - whether a book is already available in the library, and
 - whether funds are available

Example: Top-down Strategy



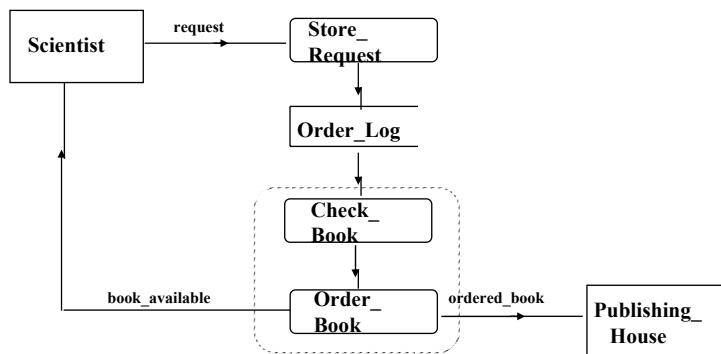
(a) Skeleton



(b) Applying primitive T1 to Handle_Request process

T1: order processing consists of checking the feasibility of a purchase, followed by compiling an order.

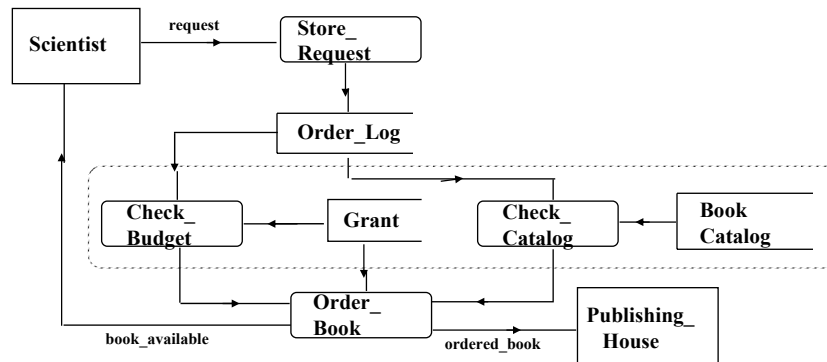
Example: Top-down Strategy (cont.)



(c) Applying primitive T2 to Process_Orders

T2: the policy of the librarian is to collect book orders and process them periodically.

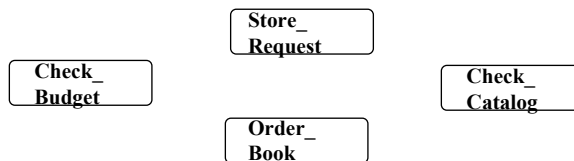
Example: Top-down Strategy (cont.)



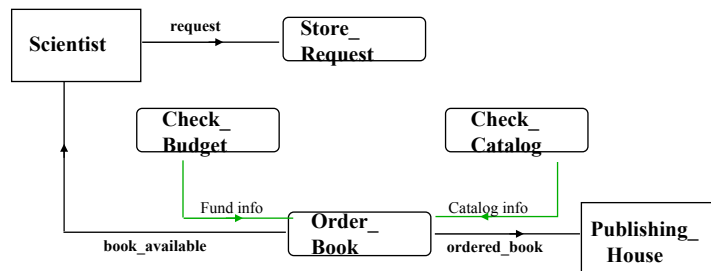
(d) Applying primitive T3 to Check_Book

T3: the check_book activity can be decomposed into two parallel activities: checking whether a book is already available in the library, and whether funds are available

Example: Bottom-up Strategy

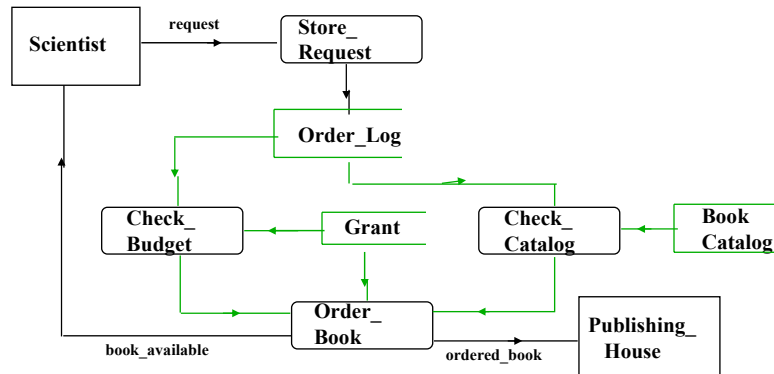


(a) initial elementary processes



(b) introduction of interfaces

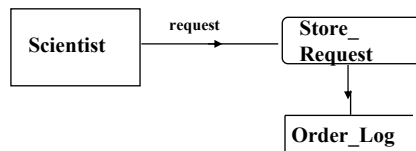
Example: Bottom-up Strategy (cont.)



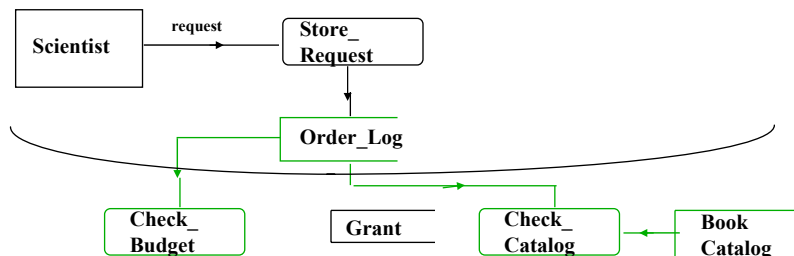
(c) introduction of data flows and data stores

Home work: put in all the missing flow names

Example: Mixed Strategy

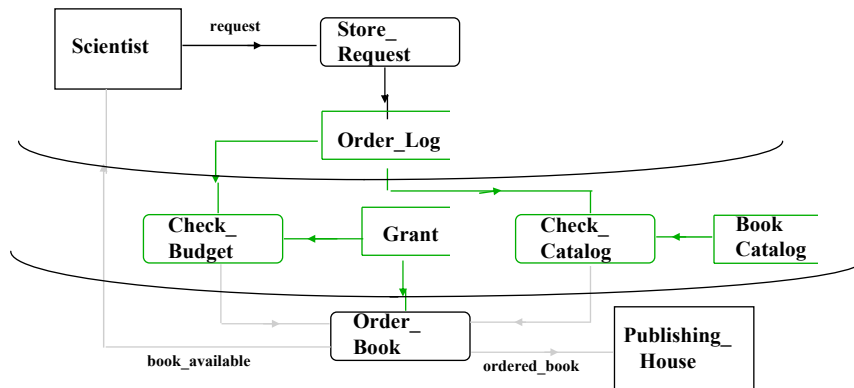


(a) First inside-out expansion



(b) Second inside-out expansion

Example: Mixed Strategy (cont.)



(c) Third inside-out expansion

Data flow diagramming mistakes

■ Illegal data flows

- One of the patterns of data flow analysis is that all flows must begin with or end at a processing step.

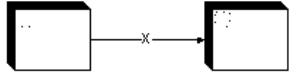
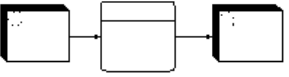
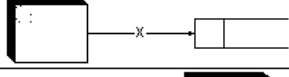



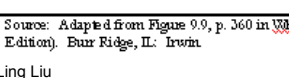
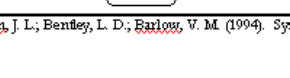
→ This makes sense, since presumably data cannot simply metastasize on its own without being processed.

- This simple rule means that the following mistakes can be fairly easily identified and corrected in a DFD.

Data flow diagramming mistakes

■ Illegal data flows

Table 2: Common data flow diagramming mistakes

Wrong	Right	Description
		A source or a sink cannot provide data to another source or sink without some processing occurring.
		Data cannot move directly from a source to a data store without being processed.
		Data cannot move directly from a data store to a sink without being processed.
		Data cannot move directly from one data store to another without being processed.

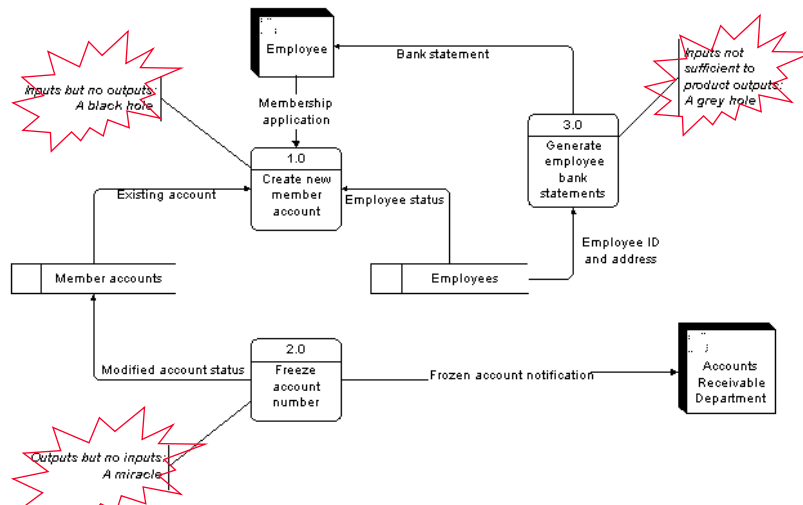
Source: Adapted from Figure 9.9, p. 360 in Whitten, J. L., Bentley, L. D., Barlow, V. M. (1994). Systems Analysis and Design Methods (Third Edition). Burr Ridge, IL: Irwin.

Data flow diagramming mistakes

- Diagramming mistakes: Black holes, grey holes, and miracles
- A second class of DFD mistakes arise when the outputs from one processing step do not match its inputs. It is not hard to list situations in which this might occur:
 - A processing step may have input flows but no output flows. This situation is sometimes called a *black hole*.
 - A processing step may have output flows but no input flows. This situation is sometimes called a *miracle*.
 - A processing step may have outputs that are greater than the sum of its inputs - e.g., its inputs could not produce the output shown. This situation is sometimes referred to as a *grey hole*.
- The following diagram illustrates these common DFD mistakes. By tracing the inputs and outputs affecting each processing step, you can avoid them in your own diagrams.

Data flow diagramming mistakes

Figure 5: Common DFD drafting errors: black holes, grey holes, miracles



Source: Adapted from Figure 9.3, p. 356 in Whitten, J. L.; Bentley, L. D.; Barrow, V. M. (1994). Systems Analysis and Design Methods (Third Edition). Burr Ridge, IL: Irwin.

Data flow diagramming mistakes

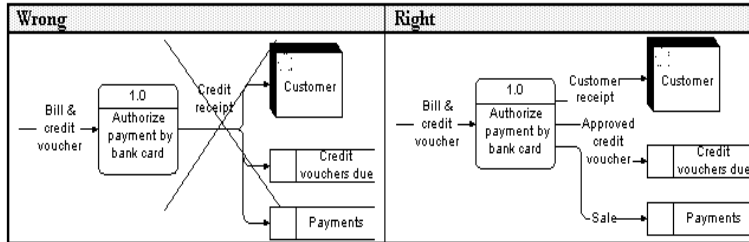
- **DFDs are not flow charts**
- Flow chart diagrams can be useful for describing programming logic or understanding a single sequence of process activities.
- It is important to recognize, however, that DFDs are *not* flow charts.
 - Flow charts often show both processing steps and data "transfer" steps (e.g., steps that do not "process" data); DFDs only show "essential" processing steps.
 - Flow charts might (indeed, often do) include arrows without labels: DFDs never show an unnamed data flow.
 - Flow charts show conditional logic; DFDs don't (the conditional decisions appear at lower levels, always within processing steps).
 - Flow charts show different steps for handling each item of data; DFDs might include several data items on a single flow arrow.

Data flow diagramming mistakes

■ DFDs are not flow charts

- With these distinctions in mind, the following diagrams suggest some DFD drafting mistakes that might be influenced by prior experience with flow charts.

Figure 7: Conditional/diverging data flows should be replaced by individual flows

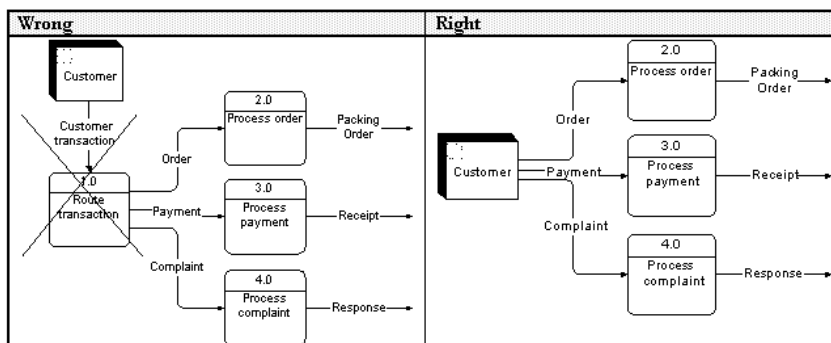


Source: Adapted from Figure 9.8, p. 359 in Whitten, J. L.; Bentley, L. D.; Barlow, V. M. (1994). *Systems Analysis and Design Methods* (Third Edition). Burr Ridge, IL: Irwin.

Data flow diagramming mistakes

- In the example above, a processing step is included that does not actually change any data. This step (titled "route transaction") might appear on a flow chart but would not appear on a DFD.

Figure 6: Processing steps that do not change data don't belong in DFDs



Source: Adapted from Figure 9.6, p. 357 in Whitten, J. L.; Bentley, L. D.; Barlow, V. M. (1994). *Systems Analysis and Design Methods* (Third Edition). Burr Ridge, IL: Irwin.

Five Processing Characteristics

- As a final aid in developing DFDs, consider the following description of processing steps. It suggests five characteristics of processing steps that change data (and so deserve to be included on a DFD).

Table 3: Processing steps to be considered in drafting DFDs

Include processing steps that	Example(s)
Perform computations	A processing step that develops charges associated with a product or service, e.g., "Tncc consulting engagement"
Make decisions	A processing step that qualifies a potential customer as a good prospect based on demographics, income level, and the number of times that the individual has responded to company product trials.
Split data flows based on content or business rules	A processing step that separates approved orders from rejected orders based on credit rules.
Filter and/or summarize data flows to produce new data flow(s)	A processing step where specific data items may not change, but the structure of the data does, e.g., filtering invoice data to identify products that were in highest demand during the past two week.

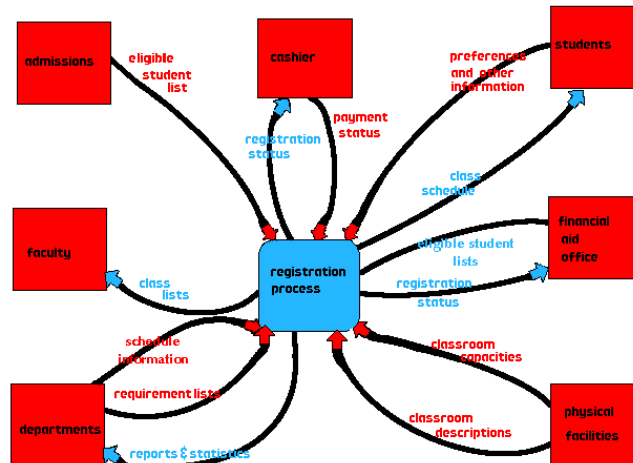
Source: Adapted from p. 355 in Whitten, J. L.; Bentley, L. D.; Barlow, V. M. (1994). *Systems Analysis and Design Methods* (Third Edition). Burr Ridge, IL: Irwin.

Data Flow Diagram Summary

- Data flow diagrams can assist in
- **Isolating the component parts of a business process**, reducing the analytical complexity involved in determining the specifications that process support software would have to meet.
- **Shifting the focus** of a process description **to the data** flows and processing steps that the process represents.
- **Identifying** data-related process characteristics that could be candidates for **process design improvements**.
- **Identifying data stores that isolate entities** that could be further developed using entity-relationship analysis.

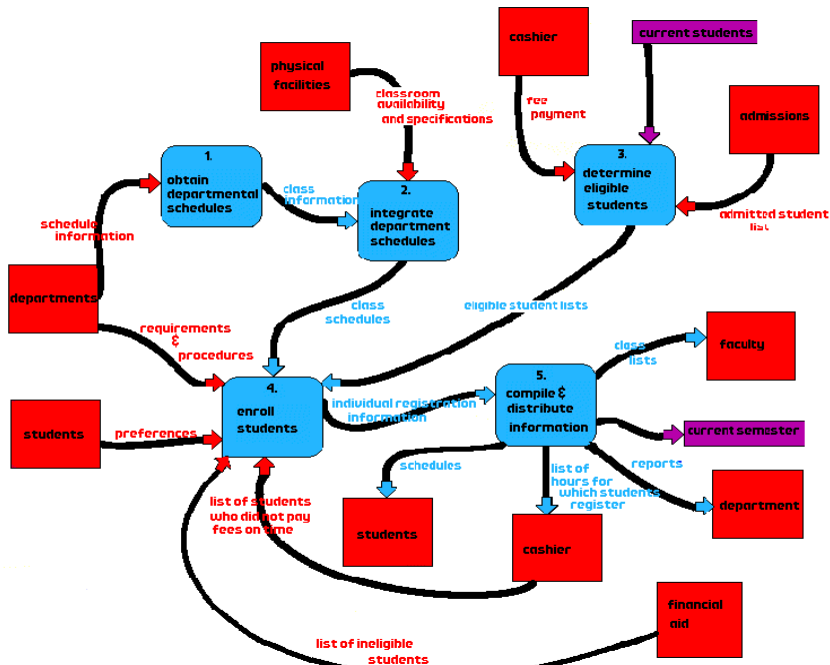
More Data Flow Diagrams Examples

- These examples demonstrate the flows of information for *the registration process*.
- The first of these is the *context diagram*.



© Ling Liu

41

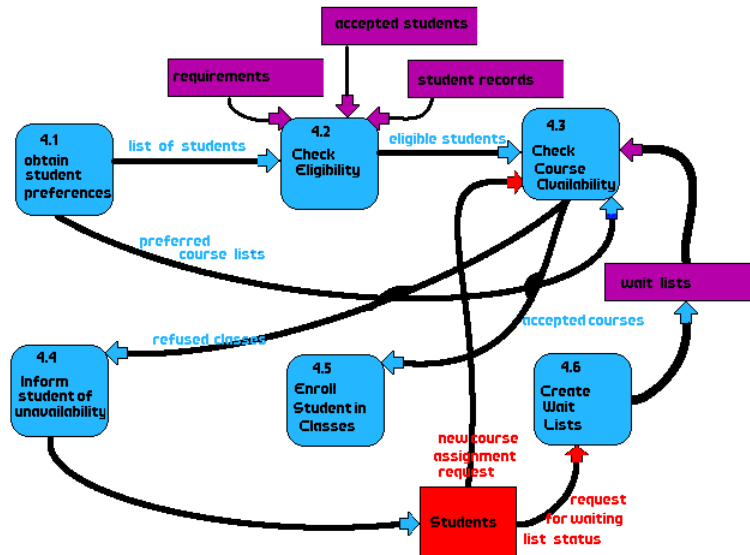


© L

42

More Data Flow Diagrams Examples

- This page illustrates the *explosion* of Process 4, Enroll



Guidelines/Gumption Traps:

- (Places where DFDing can go astray)
 1. **System boundary establishment** is an important judgment call. External entities aid in determining where the boundary is established. An interfacing system can be shown as an external entity. It may be necessary to dictate the input of the external entity to assure system control.
 2. **Label your processes carefully** and vividly. A process that is labeled "Produce Report" and has the output of "Report" tells a reviewer very little. If you have trouble labeling anything on the diagram, it often is because you do not have adequate understanding. Choose names carefully.
 3. **Think logical, not physical.** Ignore media, color, font, layout, packaging, time, sequencing, etc. Think "what", not "how". Something logical can be implemented physically in more than one way. Including "when" and "where" and "how" means you are getting physical.

Guidelines/Gumption Traps:

- **(Places where DFDing can go astray)**
- 4. **Think data, not control, flow.** Data flows are pathways for data. Think about what data is needed to perform a process or update a data store. A data flow diagram is not a flowchart and should not have loops or transfer of control. Think about the data flows, data processes, and data storage that are needed to move a data structure through a system.
- 5. **Concentrate first on what happens to a "good" transaction.** Systems people have a tendency to lose sight of the forest because they are so busy concentrating on the branches of the trees.
- 6. Reviewers will not be convinced by confusion. A quality data flow diagram will be so simple and straightforward that people will wonder what took you so long.
- 7. Data store to data store, external entity to external entity, or external entity to data store connection usually do not make sense. Data flows with an arrowhead on each end cause confusion in labeling. Do not use them.
- 8. **Do not try to put everything you know on the data flow diagram.** The diagram should serve as index and outline. The index/outline will be "fleshed out" in the data dictionary, data structure diagrams, and procedure specification techniques.
- **Good Luck, Have Fun, and Stay on those Happy Trails.....**

Questions

