Learning Objectives

- Understand the logical modeling of processes through studying data-flow diagrams
- Learn how to draw data-flow diagrams using rules and guidelines
- Learn how to decompose data-flow diagrams into lower-level diagrams
- Balancing of data-flow diagrams
- Understand Structured English
- Understand Decision Tables

Requirements Determination

- Requirement structuring has two main components:
  - Process Modeling
  - Data Modeling
What is Process Modeling?

- Process Modeling
  - Graphically representing the processes, or actions, that capture, manipulate, store and distribute data between a system and its environment and among system components

- Data-flow Diagram (DFD)
  - One of the main process modeling tools
  - Graphically illustrates movement of data between external entities and the processes and data stores within a system

Deliverables and Outcomes of Requirements Structuring

- Context data-flow diagram (DFD)
  - Shows the system's scope indicating element inside & outside the system

- DFDs of current system
  - Illustrates current processes & technologies used to transform inputs into outputs
  - Enable analysts to understand current system

- DFDs of new logical system
  - Technology independent
  - Show data flows, structure and functional requirements of new system

- Project dictionary and CASE repository
  - Contain entries for objects in all diagrams
  - The documentation component

DFD Symbols


  - Drawn as an arrow
  - Depicts data that are in motion and moving as a unit from one place to another in the system
  - Select a meaningful name to represent the data

  - Drawn as a rectangle with the right vertical line missing
  - Depicts data at rest
  - May represent data in file folder, computer-based file, notebook
  - Label includes name of the store as well as the number
### DFD Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![External Entity](source:external.png) | - Drawn as a square symbol  
- Depicts the origin and/or destination of data  
- Also referred to as an external entity  
- Name states what the external agent is  
- Because they are external, many of their characteristics are not of interest to us |

| ![Process](source:process.png) | - Drawn as a rectangle with rounded corners  
- Depicts work or actions performed on data so that they are transformed, stored, or distributed  
- Number of process as well as name are recorded |

### Process Modeling Steps

- **Purpose:** converting gathered requirements into diagrams using a systematic method

- **Step 1:** Draw the Context Diagram  
  - A data-flow diagram (DFD) of the scope of an organizational system that shows the system boundaries, external entities that interact with the system and the major information flows between the entities and the system

- **Step 2:** Identifying process(es) embedded in the single process of the context diagram  
  - Your understanding of the gathered requirements helps you list the major processes/actions taking place in existing system or that should take place in the new system  
  - You may also identify some data stores along the processes

- **Step 3:** Draw Level-0 Diagram  
  - Use identified processes to draw the data-flow diagram (DFD) that represents a system’s major processes, data flows, and data stores at a higher level. It’s called Level-0 DFD

  See example on next slide
Process Modeling Steps (cont)

1. Step 4: Decompose Level-0 DFD processes
   - For each Level-0 process:
     - Identify the subprocesses that need to be performed to complete the process.
     - Identify key data stores along the identified subprocesses.
     - Draw the process' Level-1 diagram.

Q: This figure shows the decomposition of the 1.0 Level-0 process. How many subprocesses were found embedded in the 1.0 process?

Note: Decomposition of all Level-0 processes need to be completed (as well as drawing all corresponding Level-1 diagrams) before decomposing Level-1 processes.
Process Modeling Steps (cont)

- **Step 5:** Decompose Level-1 DFD processes
  - For each Level-1 process:
    - Identify the subprocesses that need to be performed to complete the process
    - Identify key data stores along the identified subprocesses
    - Draw the process' Level-2 diagram

Data-Flow Diagramming Rules

- Basic rules that apply to all DFDs:
  - Inputs to a process are always different than outputs
  - Objects always have a **unique name**
    - In order to keep the diagram uncluttered, you can repeat data stores and data flows on a diagram

Data-Flow Diagramming Rules (cont)

- **Coupling Principle**
  - Processes 1.0 and 3.0 are linked by a data flow. They are **coupled**. The coupling suggests that whenever 1.0 generates a data flow, 3.0 must be ready to take it. => Timing must be taken into consideration.
  - D2 is a buffer data store sitting between processes 2.0 and 4.0. This makes them **decoupled** processes. => Timing is not an issue to take into consideration.
Data-Flow Diagramming Rules (cont)

- **Process**
  - A. No process can have only outputs (a miracle)
  - B. No process can have only inputs (black hole)
  - C. A process has a verb phrase label

- **Data Store**
  - D. Data cannot be moved from one store to another data store
  - E. Data cannot move from an outside source to a data store
  - F. Data cannot move directly from a data store to a data sink
  - G. Data store has a noun phrase label
Data-Flow Diagramming Rules (cont)

- **Source/Sink**
  - **H.** Data cannot move directly from a source to a sink
  - **I.** A source/sink has a noun phrase label

---

Data-Flow Diagramming Rules (cont)

- **Data Flow**
  - **J.** A data flow has only one direction of flow between symbols
  - **K.** A fork means that exactly the same data go from a common location to two or more processes, data stores, or sources/sinks

---

Data-Flow Diagramming Rules (cont)

- **Data Flow (Continued)**
  - **L.** A join means that exactly the same data come from any two or more different processes, data stores or sources/sinks to a common location
  - **M.** A data flow cannot go directly back to the same process it leaves
  - **N.** A data flow to a data store means update
  - **O.** A data flow from a data store means retrieve or use
  - **P.** A data flow has a noun phrase label
Balancing DFDs Rules

- When decomposing a DFD, you must conserve inputs to and outputs from a process at the next level of decomposition.
- This is called balancing.

Example: In Figure 5-4, notice that:
- There is 1 input to the system.
- Three outputs:
  - Customer receipt
  - Food order
  - Management reports

Balancing DFDs Rules (cont)

Example (Continued):
- Notice Figure 5-5. We have the same inputs and outputs to the system.
- We can say that the context diagram and level-0 DFD are balanced.

Guidelines for Drawing DFDs

1. Timing
   - Time is not represented well on DFDs.
   - No indication of whether data flow occurs 1 or 10 times a day, a week, etc.
   - Best to draw DFDs as if the system has never started and will never stop.

2. Iterative Development
   - Analyst should expect to redraw diagram several times before reaching the closest approximation to the system being modeled.

3. Primitive DFDs
   - Lowest logical level of decomposition.
   - Decision has to be made when to stop decomposition.
Rules for stopping decomposition

- When each process has been reduced to a single decision, calculation or database operation
- When each data store represents data about a single entity
- When the system user does not care to see any more detail
- When every data flow does not need to be split further to show that data are handled in various ways
- When you believe that you have shown each business form or transaction, online display and report as a single data flow
- When you believe that there is a separate process for each choice on all lowest-level menu options

Logic Modeling

- DFDs do not show the logic inside the processes
- Logic modeling involves representing internal structure and functionality of processes depicted on a DFD
- Two methods:
  - Structured English
  - Decision Tables

Structured English

- Modified form of English used to specify the logic of information processes
- Uses a subset of English
  - Action verbs (Read, update, sort, generate..)
  - Noun phrases
  - No adjectives or adverbs
- No specific standards
Structured English (continued)

- Similar to programming language
  - If conditions
    ```plaintext
    BEGIN IF
      IF Quantity is less than Minimum-order-quantity
      THEN GENERATE new order
      ELSE DO nothing
    END IF
    ```
  - Case statements

Figure 5-15 shows structured English representation for Hoosier Burger

Decision Tables

- A matrix representation of the logic of a decision
- Specifies the possible conditions and the resulting actions
- Best used for complicated decision logic