LECTURE 11: PROCESS MODELING

Outline

- Logical modeling of processes
- Data Flow Diagram Elements
- Functional decomposition
- Data Flows
- Rules and Guidelines
- Structured Analysis with Use Cases

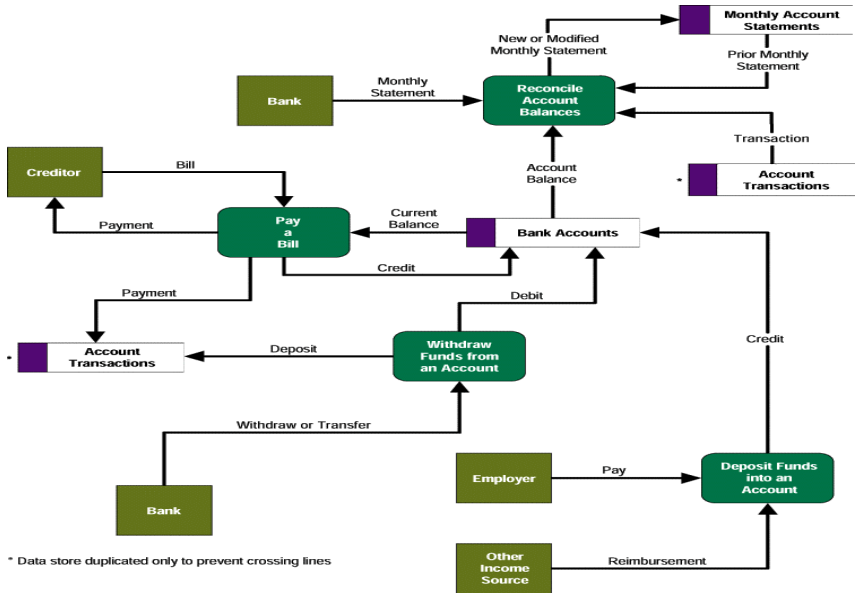
Learning Objectives

- Understand the logical modeling of processes through studying data flow diagrams
- How to draw data flow diagrams using rules and guidelines
- How to decompose data flow diagrams into lower-level diagrams
- Balancing of data flow diagrams
- Discuss the use of data flow diagrams as analysis tools

Process Modeling

- Graphically represent the processes that capture, manipulate, store and distribute data between a system and its environment and among system components
- Data flow diagrams (DFD)
    - Graphically illustrate movement of data between external entities and the processes and data stores within a system
- Modeling a system's process
    - Utilize information gathered during requirements determination
    - Structure of the data is also modeled in addition to the processes
- Deliverables and Outcomes
    - Set of coherent, interrelated data flow diagrams
    - Context data flow diagram (DFD)
        - Scope of system
    - DFDs of current system
        - Enables 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

Simple Data Flow Diagram



1. DFD Elements

Data Flows & Control Flows

- A data flow represents an input of data to a process, or the output of data from a process.
  - A data flow may also be used to represent the creation, reading, deletion, or updating of data in a file or database (data store).
  - A composite data flow is a data flow that consists of other data flows.
- A control flow represents a condition or nondata event that triggers a process.
  - Used sparingly on DFDs.

Data Stores

- A data store is an inventory of data.
  - Frequently implemented as a file or database.
  - A data store is "data at rest" compared to a data flow that is "data in motion."
  - Almost always one of the following:
    - Persons (or groups of persons)
    - Places
    - Objects
    - Events (about which data is captured)
    - Concepts (about which data is important)

  - Data stores depicted on a DFD store all instances of data entities (depicted on an ERD)

External Agents

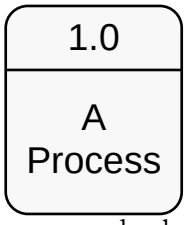
- An external agent defines a person, organization unit, or other organization that lies outside of the scope of the project but that interacts with the system being studied.
  - External agents define the "boundary" or scope of a system being modeled.
  - As scope changes, external agents can become processes, and vice versa.

- Almost always one of the following:
  - Office, department, division inside the business but outside the system scope.
  - An external organization or agency.
  - Another business or another information system.
  - One of your system's end-users or managers

- Alternatively: Source/Sink
  - Depicts the origin and/or destination of the data
  - Sometimes referred to as an external entity
  - Drawn as a square symbol
  - Name states what the external agent is
  - Because they are external, many characteristics are not of interest to us

Process Concepts

- A **process** is work performed on, or in response to, incoming data flows or conditions so that they are transformed, stored or distributed
- **A System <u>is</u> a Process**



2. Process Decomposition

Functional Decomposition Decomposition

- The act of breaking a system into its component subsystems, processes, and subprocesses. Each level of abstraction reveals more or less detail.

Decomposition Diagrams

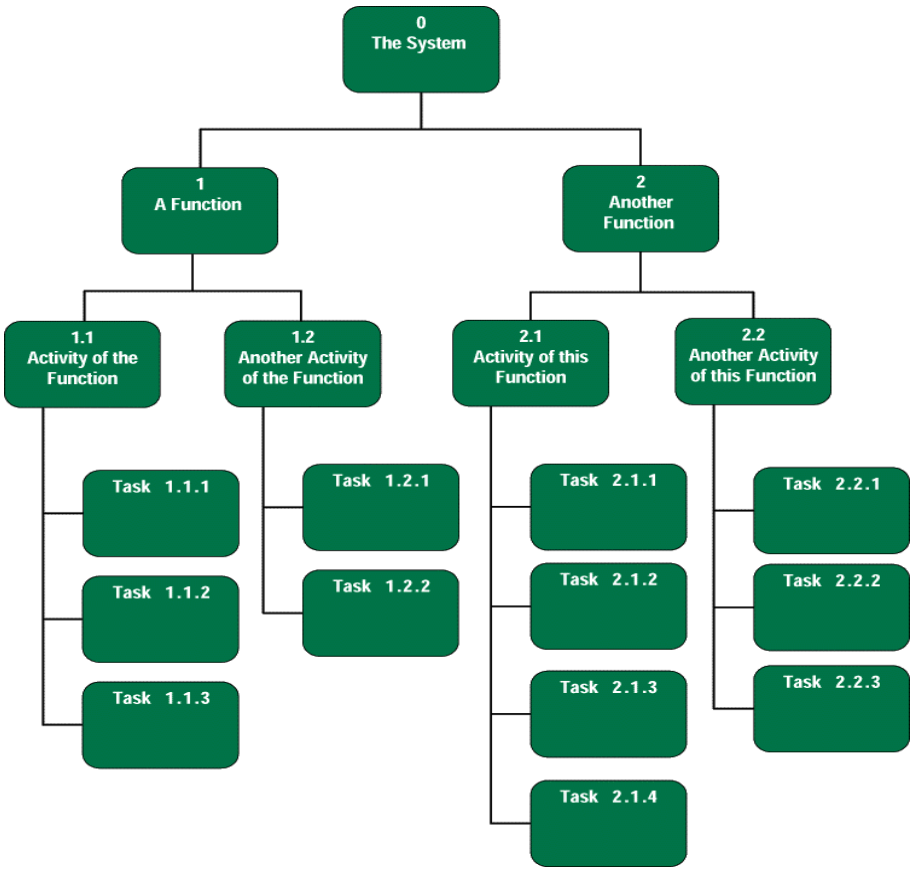
- A decomposition diagram or hierarchy chart shows the top-down, functional decomposition of a system
- Numbering Scheme

Decomposition of DFDs

- Functional decomposition
  - Act of going from one single system to many component processes
  - Repetitive procedure
  - Lowest level is called a primitive DFD
- Level-N Diagrams
  - A DFD that is the result of *n* nested decompositions of a series of subprocesses from a process on a level-0 diagram

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
- Shows the context into which the business process fits
- Shows the overall business process as just *one* process
- Shows all the outside entities that receive information from or contribute information to the system



Level 0 Diagram

- A data flow diagrams (DFD) that represents a system's major processes, data flows and data stores at a higher level
- Shows all the processes that comprise the overall system
- Shows how information moves from and to each process
- Adds data stores

Level 1 Diagrams

- Shows all the processes that comprise a single process on the level 0 diagram
- Shows how information moves from and to each of these processes
- Shows in more detail the content of higher level process
- Level 1 diagrams may not be needed for all level 0 processes

Level 2 Diagrams

- Shows all processes that comprise a single process on the level 1 diagram
- Shows how information moves from and to each of these processes
- Level 2 diagrams may not be needed for all level 1 processes
- Correctly numbering each process helps the user understand where the process fits into the overall system

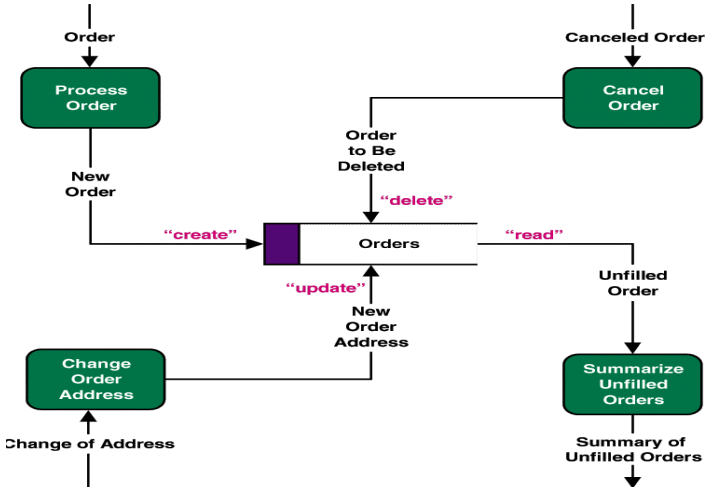
Rules for stopping decomposition

- 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, on-line 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

3. Data Flows

Data Flows to and from Data Stores



Diverging and Converging Data Flows

- A diverging data flow is one that splits into multiple data flows.
  - Useful for illustrating data that starts out naturally as one flow, but needs to be routed to parallel processes.
  - Also useful for illustrating multiple copies of the same output going to different destinations.
- A converging data flow is the merger of multiple data flows into a single packet.
  - Useful for illustrating data from multiple sources that must come back together for some subsequent processing

Alternative Data Flows

- Where a process can produce different data given different conditions
- We show both data flows and use the process description to explain why they are alternatives
- Tip -- alternative data flows often accompany processes with IF statements

4. Rules and Guidelines

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

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

- When decomposing a DFD, you must conserve inputs to and outputs from a process at the next level of decomposition
- This is called balancing
- We can split a data flow into separate data flows on a lower level diagram

Four Additional Advanced Rules Due to Balancing DFDs

Q. A composite data flow on one level can be split into component data flows at the next level, but no new data can be added and all data in the composite must be accounted for in one or more subflows.

R. The input to a process must be sufficient to produce the outputs (including data placed in data stores) from the process. Thus all outputs can be produced, and all data in inputs move somewhere, either to another process or to a data store outside the process or on a more detailed DFD showing a decomposition of that process.

S. At the lowest level of DFDs, new data flows can be added to represent data that are transmitted under exceptional conditions; these data flows typically represent error messages or confirmation notices.

T. To avoid having data flow lines cross each other, you may repeat data store, source/sink on a DFD. Use an additional symbol, like a double line on the middle vertical line of a data

store symbol, or a diagonal line of a sink/source square, to indicate repeated symbol.

5. Creating DFDs From Use Cases

Integrating Scenario Descriptions

- DFDs start with the use cases and requirements definition
- Generally, the DFDs integrate the use cases
- Names of use cases become processes
- Inputs and outputs become data flows
- "Small" data inputs and outputs are combined into a single flow

Steps in Building DFDs

- Build the context diagram
- Create DFD fragments for each use case
- Organize DFD fragments into level 0 diagram
- Decompose level 0 processes into level 1 diagrams as needed; decompose level 1 processes into level 2 diagrams as needed; etc.
- Validate DFDs with user to ensure completeness and correctness

Build the Context Diagram

- Draw one process representing the entire system (process 0)
- Find all inputs and outputs listed at the top of the use cases that come from or go to external entities; draw as data flows
- Draw in external entities as the source or destination of the data flows

Creating DFD Fragments

- Each use case is converted into one DFD fragment
- Number the process the same as the use case number
- Change process name into verb phrase
- Design the processes from the viewpoint of the organization running the system
- Add data flows to show use to data stores as sources and destinations of data
- Layouts typically place
    - processes in the center
    - inputs from the left
    - outputs to the right
    - stores beneath the processes

Creating the Level 0 Diagram

- Combine the set of DFD fragments into one diagram
- Generally move from top to bottom, left to right
- Minimize crossed lines
- Iterate as needed
    - DFDs are often drawn many times before being finished, even with very experienced systems analysts

Creating Level 1 Diagrams (and Below)

- Each use case is turned into its own DFD

- Take the steps listed on the use case and depict each as a process on the level 1 DFD
- Inputs and outputs listed on use case become data flows on DFD
- Include sources and destinations of data flows to processes and stores within the DFD
- May also include external entities for clarity
- Input data flows shown on a parent DFD are often unbundled on the child diagram using splits
- Output data flows shown on a child DFD are often bundled using joins and shown as a larger data flow on the parent diagram
- When to stop decomposing DFDs?
    - Ideally, a DFD has at least 3 processes and no more than 7-9.