Example system:
- a generic warehouse-inventory information system

Levelling Data Flow Diagrams
- any “real” system is too large to represent as a single data flow diagram
- the solution is to decompose the system into a hierarchy of levels of processing
- the process model of the system then consists of a set of levelled data flow diagrams
- levelling of DFDs improves their readability and usefulness as a communication tool

Levelling of DFDs
Levelling creates a hierarchical decomposition of the processing within the system
- Context diagram
  - Level 0 diagram
    - Level 1 diagrams
    - Level 2 diagrams
    - Level n diagrams

Context Diagram
- the highest level data flow diagram is the context diagram
- the context diagram shows the interaction of the system with its environment in terms of data flows
- the context diagram defines the boundary of the system (the scope of the system)
- only the data flows which leave the system and the data flows which come from outside the system are shown
Example Context Diagram

Level Zero Diagram

- all external agents (sources and sinks) are included because the level zero diagram, like the context diagram, represents the entire system

- the number of each process ends in .0 which corresponds to the level of the diagram:
e.g. 1.0, 2.0, 3.0 etc.
often just 1, 2, 3 etc. are used

Level Zero Diagram

- the level zero data flow diagram is the diagram at the level immediately below the context diagram

- it "expands" the single process on the context diagram to show the major, high-level processes (or functions) within the system

Example Level Zero Diagram

Level 1 diagrams

- a set of data flow diagrams is created at Level 1
- there is one Level 1 diagram for each of the processes at Level zero
- each diagram decomposes a Level zero process into several processes
Level 1 and lower level diagrams

- each Level 1 diagram and diagrams at lower levels (e.g. Level 2, Level 3) show only a part of the processing
- Level 1 processes are numbered 1.1, 1.2, 1.3, and 2.1, 2.2, 2.3 etc
- that part is shown in more detail than on the Level zero diagram
- no external agents are shown on a Level 1 or lower diagrams, as the entire system is not being represented

Example Level zero diagram

Example Level 1 diagram

Guidelines for Levelling DFDs

External communication:
- external agents represent entities in the environment of our information system
- external agents are outside the scope of our information system
- we do NOT model interactions between external agents
- we do NOT allow external agents to interact directly with data stores

Guidelines for Levelling DFDs

- numbering:
  when a process is decomposed, its diagram is given the same number as that process
- balancing of levelled DFDs:
  all data flows entering and leaving a process must appear on the corresponding diagram which decomposes that process
- external agents:
  are only included on the two diagrams which represent the entire system, i.e. the context and level zero diagrams
Guidelines for Levelling DFDs

the access to data stores across levels of diagrams must be consistent:
- the direction of accesses must match and all accesses on higher level diagrams must appear on corresponding lower level diagrams
- a data store is first shown on the highest level diagram where it is accessed by more than one process
- it can then appear on all lower level diagrams where it is accessed

Guidelines for Levelling DFDs

How many levels should be in a set of DFDs?
- each diagram usually has between 3 and 7 processes
- level the diagrams until bottom level or primitive processes are reached:
  
  primitive processes have only 1 or 2 inputs and outputs, and cannot be further decomposed as a data flow diagram

Guidelines for Levelling DFDs

- partition processes to minimise the data flows between them
- partition processes to form cohesive, related groups of activities
- not all parts of the system may need to be decomposed to the same level

An example - Context Diagram

An example - Level zero diagram
Logical and physical DFDs

Models may focus on either:

- the "physical" view of the real world – how things are done
  OR
- the "logical" view of the real world – what things are done

Physical DFDs

- represent a particular way of implementing the processes and data in a system
- they are technology dependent – they specify particular methods of doing tasks
- they show how the processing takes place and how the data is implemented

Logical DFDs

- represent what a system must do regardless of how it is implemented
- they are technology independent
- they show what processing, data movements and data storage must occur in a system
- they show the essential aspects of a system

Using Logical and Physical DFDs

- Physical DFDs modelling current system: help systems analysts become familiar with how a business or system operates
- Physical DFDs modelling new systems: model the technical and human design decisions to be implemented
- users can relate to physical DFDs more readily because they contain implementation details: landmarks e.g. people or roles, actual locations
Use of Logical and Physical DFDs

Systems analysts often begin with physical DFDs of current systems:

• convert that physical DFD to a logical model in order to focus on essential elements
• use the logical model to model a new logical solution (DFD)
• Convert the logical solution into a physical DFD (implementation) model
• implementation details can be removed from physical DFDs

Physical to Logical DFDs

• use names for data flows and data stores which indicate their content, not their physical form or location
• use names for processes that indicate what, not how

Physical to Logical DFDs

<table>
<thead>
<tr>
<th>Physical to Logical DFDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>AZ104 form</td>
</tr>
<tr>
<td>sales order</td>
</tr>
</tbody>
</table>

Logical and Physical DFDs

<table>
<thead>
<tr>
<th>Physical DFDs</th>
<th>Logical DFDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>How processing is implemented</td>
</tr>
<tr>
<td>Processes</td>
<td>Actual sequence</td>
</tr>
<tr>
<td>Naming</td>
<td>Forms, locations, people/roles</td>
</tr>
<tr>
<td>Data flows</td>
<td>Detailed/ specific/ duplicated data describing exact implementation</td>
</tr>
</tbody>
</table>

Function Decomposition: a Process Modelling Technique

• function decomposition is the decomposing of a system into its component functions and processes as a way of managing complexity
• function decomposition creates a top-down view: it shows a hierarchy of levels of increasingly detailed processes within a system
• a function decomposition model (or diagram) is used to represent the hierarchical decomposition and structure of the processes of a system
• the decomposition of functions corresponds to levelling of processes in DFDs

Function Decomposition Diagrams

• a function is a high-level set of related activities that are ongoing: a function is a broad, generic activity
• a process is a lower level activity that is repeatedly carried out
• functions consist of groups of related processes
• the depth and scope of function decomposition diagrams depend on the size and complexity of the system represented
Function Decomposition Diagrams

- Function decomposition diagrams showing only higher levels can be built early in systems analysis.
- More detailed process decomposition can be carried out as the system is studied in more detail.
- Function decomposition diagrams show the structure of functions and processes within the system.
- This structure may not correspond to the structure of the organisation shown in an organisation chart.


Example Function Decomposition Diagram

1. Sell Products
2. Manage Inventory
3. Control Finance

2.1 Deliver Product
2.2 Accept Delivery
2.3 Check Stock levels

Functions and processes are further decomposed.

Function Decomposition

- Function decomposition diagrams are an alternative representation of the hierarchy of functions and processes within a system.
- They may be built using either a top down or a bottom up approach.
- They provide a useful overview of the processing within a system.

References