

Data Flow Diagram

Introduction

The three most important modeling techniques used in analysing and building information systems are: **Data Flow Diagramming (DFDs)**, **Logical Data Structure modelling (LDSs)**, and **Entity Life Histories (ELHs)**

Data Flow Diagrams (DFDs) model *events* and *processes* (i.e. activities which transform data) within a system. DFDs examine *how data flows* into, out of, and within the system. **Logical Data Structures (LDSs)** represent a system's information and data in another way. LDSs map the underlying data structures as entity types, entity attributes, and the relationships between the entities

Entity Life Histories (ELHs) describe the changes which happen to 'things' (data) within the system.

These three techniques are common to many methodologies and are widely used in system analysis. Notation and graphics style may vary across methodologies, but the underlying principles are generally the same.

In SSADM (Structured Systems Analysis and Design Methodology) - which has for a number of years been widely used in the UK - systems analysts and modelers use the above techniques to build up **three, inter-related, views of the target system**, which are cross-checked for consistency.

Data Flow Diagrams (DFDs)

- [DFD Principles](#)
- [Basic DFD Notations](#)
- [\(SSADM\) DFD Notations](#)
- [DFD Levels](#)

"... a structured, diagrammatic technique for showing the functions performed by a system and the data flowing into, out of, and within it .."

Another way of looking at it is that, in SSADM, DFDs are used to answer the following data-oriented questions about a target system:

***What processing is done? When? How? Where? By whom?**
What data is needed? By whom? for what? When?*

However, we are not interested, here, in the development process in detail, only in the general modeling technique. Essentially, DFDs describe the information flows within a system.

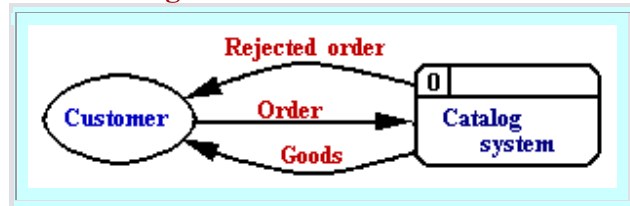
DFD Principles

- The general principle in Data Flow Diagramming is that a system can be decomposed into subsystems, and subsystems can be decomposed into lower level subsystems, and so on.
- Each subsystem represents a process or activity in which data is processed. At the lowest level, processes can no longer be decomposed.
- Each 'process' (and from now on, by 'process' we mean subsystem and activity) in a DFD has the characteristics of a system.
- Just as a system must have input and output (if it is not dead), so a process must have input and output.
- Data enters the system from the environment; data flows between processes within the system; and data is produced as output from the system

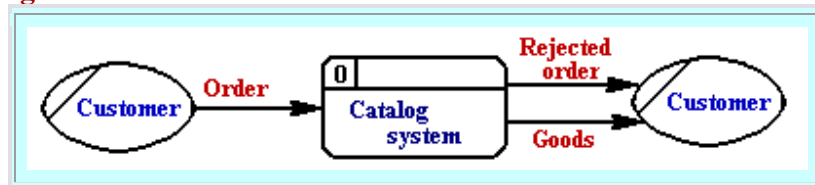
An example:

The '*Context Diagram*' is an overall, simplified, view of the target system, which contains only one process box, and the primary inputs and outputs.

Context diagram 1



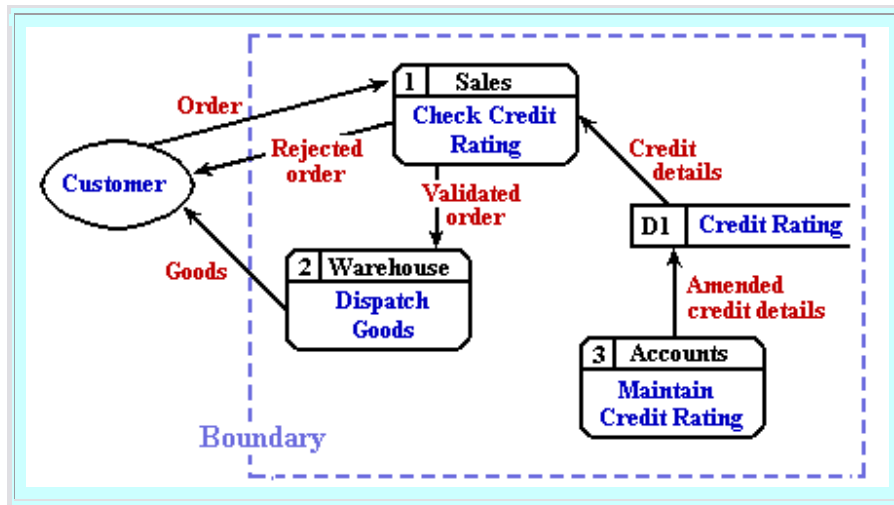
Context diagram 2



Both the above diagrams say the same thing. The second makes use of the possibility in SSADM of including duplicate objects. (In *context diagram 2* the duplication of the *Customer* object is shown by the line at the left hand side. Drawing the diagram in this way emphasizes the *Input-Output* properties of a system. See also '[Notes](#)' below)

The Context diagram above, and the decomposition which follows, are a first attempt at describing part of a 'Home Catalogue' sales system. In the modeling process it is likely that diagrams will be reworked and amended many times - until all parties are satisfied with the resulting model. A model can usefully be described as **a co-ordinated set of diagrams**.

The Top (1st level) DFD



The *Top* or *1st level* DFD, describes the whole of the target system. It **'bounds'** the system under consideration.
 (To simplify the diagram some notation has been left out - see [Notes](#) below)

Data Flow Diagrams show:

- the processes within the system
- the data stores (files) supporting the system's operation
- the information flows within the system
- the system boundary
- interactions with external entities

DFD Notations

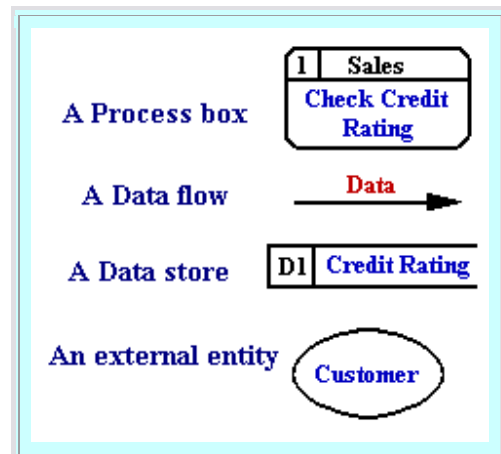
DFDs are used in most system analysis [methodologies](#).

Processes, in other methodologies, may be called 'Activities', 'Actions', 'Procedures', 'Subsystems' etc.

They may be shown as a circle, an oval, or (typically) a rectangular box.

Data are generally shown as arrows coming to, or going from the edge of a process box.

(**Note** that there is no 'Decision' symbol. A decision is a Process.)



General Data Flow Rules

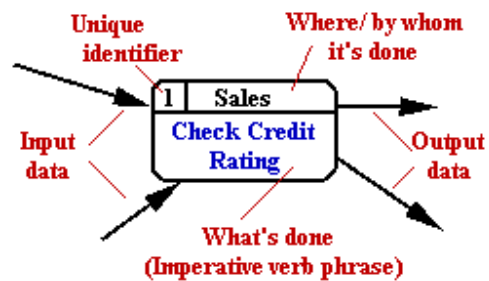
1. Entities are either 'sources of' or 'sinks' for data input and outputs - i.e. they are the originators or terminators for data flows.
2. Data flows from Entities must flow into Processes
3. Data flows to Entities must come from Processes
4. Processes and Data Stores must have both inputs and outputs (What goes in must come out!)
5. Inputs to Data Stores only come from Processes.
6. Outputs from Data Stores only go to Processes.

The Process Symbol

Processes transform or manipulate data.

Each box has a unique number as identifier (top left) and a unique name (an imperative - e.g. 'do this' - statement in the main box area) The top line is used for the location of, or the people responsible for, the process.

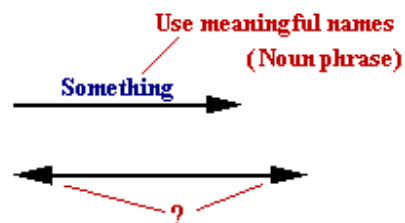
Processes are 'black boxes' - we don't know what is in them until they are decomposed
Processes transform or manipulate **input** data to produce **output** data. *Except in rare cases, you can't have one without the other.*



Data Flows depict data/information flowing to or from a process. The arrows must either start and/or end at a process box. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly.

Arrows must be named.

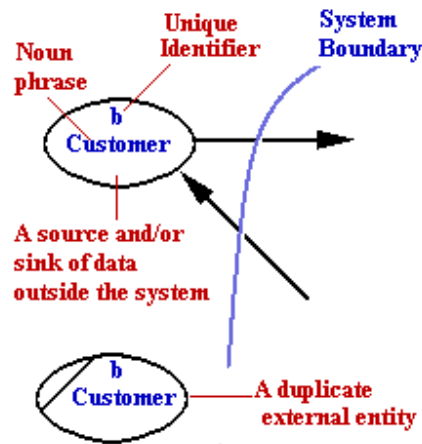
Double ended arrows may be used with care .



External Entities, also known as 'External sources/recipients, are things (eg: people, machines, organisations etc.) which contribute data or information to the system or which receive data/information from it. The name given to an external entity represents a *Type* not a specific instance of the type.

When modelling complex systems, each external entity in a DFD will be given a unique identifier.

It is common practice to have duplicates of external entities in order to avoid crossing lines, or just to make a diagram more readable.



Data Stores are some location where data is held temporarily or permanently.

In physical DFDs there can be 4 types.

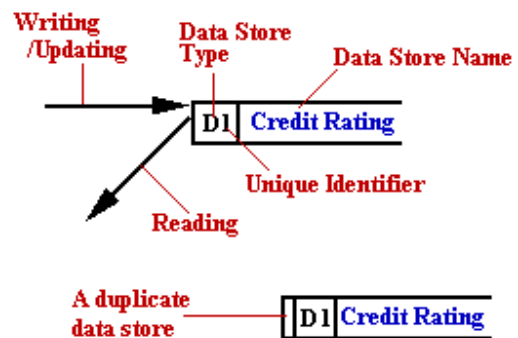
D = computerised Data

M = Manual, e.g. filing cabinet.

T = Transient data file, e.g. temporary program file

T(M) = Transient Manual, e.g. in-tray, mail box.

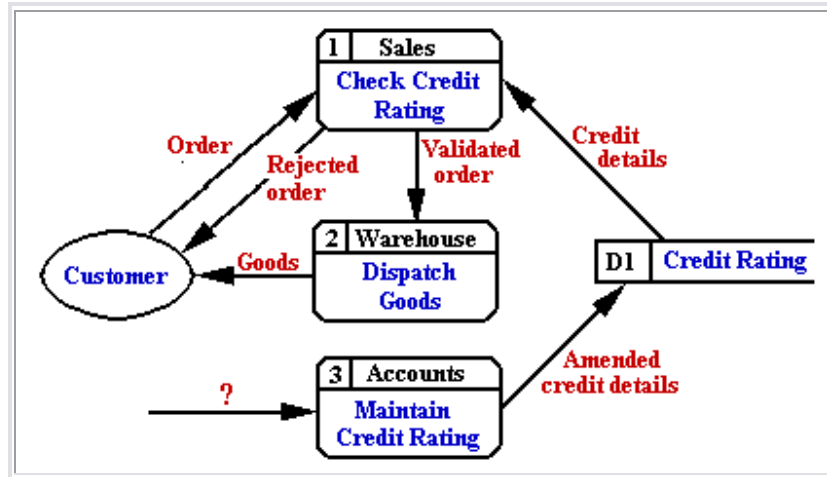
As with external entities, it is common practice to have duplicates of data stores to make a diagram less cluttered.



DFD Levels

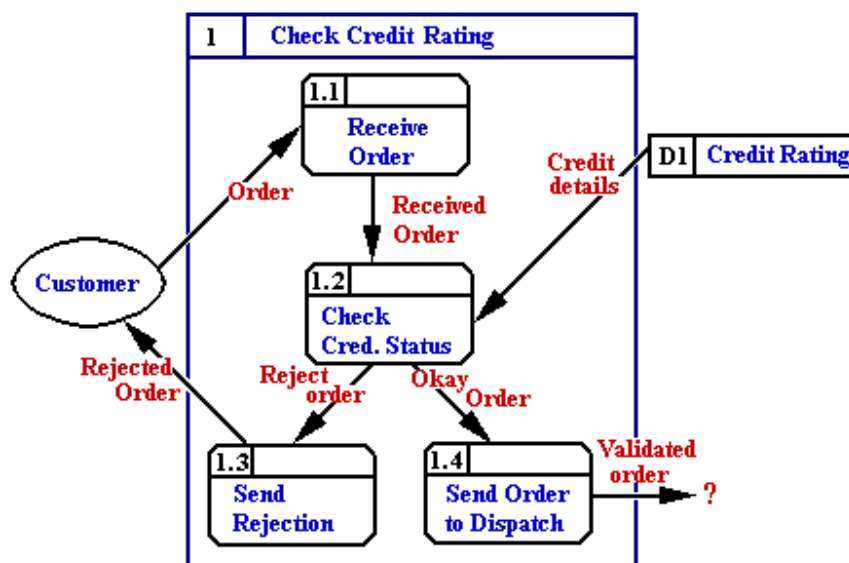
The Context and Top Level diagrams in the [example](#) start to describe 'Home Catalogue' type sales system. The two diagrams are just the first steps in creating a **model** of the system. (By model we mean a co-ordinated set of diagrams which describe the target system and provide answers to questions we need to ask about that system). As suggested the diagrams presented in the example will be reworked and amended many times - until all parties are satisfied. But the two diagrams by themselves are not enough; they only provide a high level description. On the other hand, the initial diagrams do start to break down, decompose, what might be quite a complex system into manageable parts.

A revision of the example Top Level DFD



The next step - the Next Level(s)

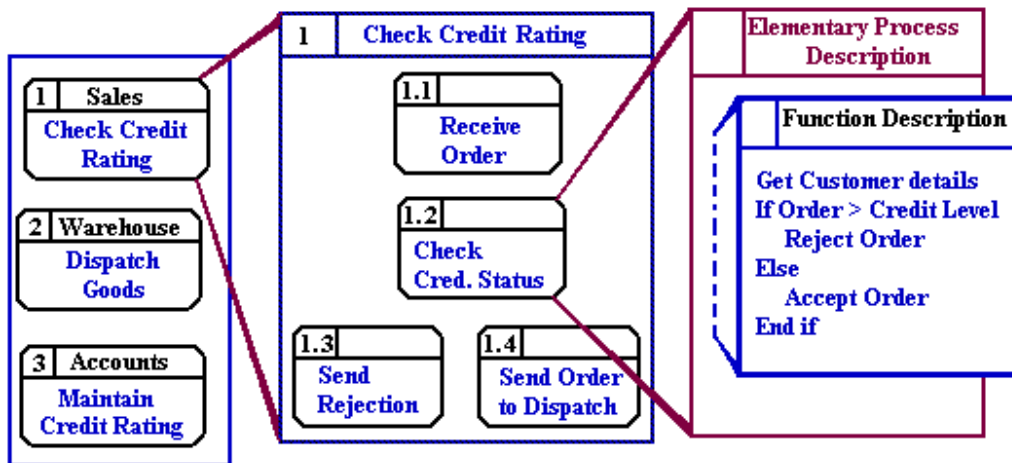
Each Process box in the Top Level diagram will itself be made up of a number of processes, and will need to be decomposed as a second level diagram.



Each box in a diagram has an identification number derived from the parent - in the top left corner. (The Context level is seen as box 0)

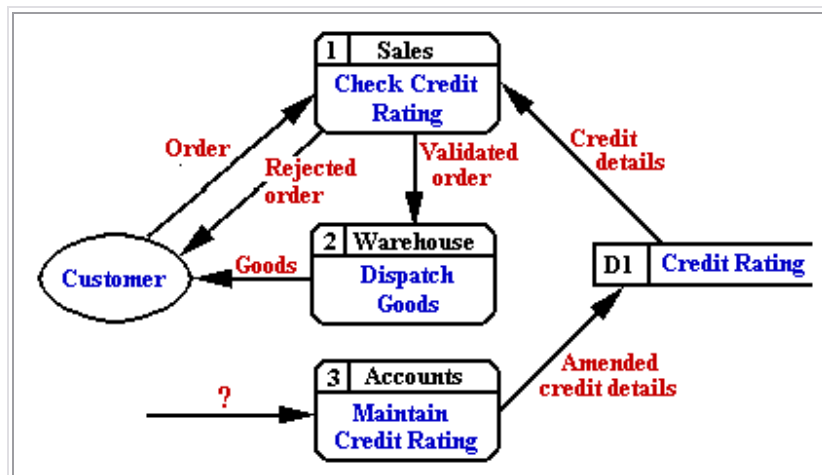
Any box in the second level decomposition may be decomposed to a third and then a fourth level. Very complex systems may possibly require decomposition of some boxes to further levels.

Decomposition stops when a process box can be described with an *Elementary Process Description* using ordinary English, later on the process will be described more formally as a *Function Description* using, for example, pseudocode.



See also: [Procedure Definitions - low level DFDs](#)

Notes



- Redrawing the diagram makes it clear that Process 3, 'Maintain Credit Rating' requires some input - if it is to produce output.
- Note that 'Goods', while it is in reality a physical thing, is seen here as *data*. This is because this is a model. We will represent 'Goods' in our model by some description. In the model, 'Goods' becomes a set of data items. In the real-world, there will be some physical objects, but in our model we only have an abstract description.

SSADM uses different sets of Data Flow Diagram to describe the target system in different ways, moving from analysis of the current system to specification of the required system:

WHAT the system does - Current Physical DFD
HOW it does it - Current Logical DFD
WHAT it should do - Required Logical DFD
HOW it should do it - Required Physical DFD

References

C.Ashworth & M.Goodland (1990) '*SSADM A Practical Approach*', McGraw-Hill.
D.E.Avison & G.Fitzgerald (1991) '*Information Systems Development*', Blackwell.
David A. Marca (1988), '*SADT. Structured Analysis and Design Technique*', McGraw-Hill.
Philip L. Weaver (1993) '*Practical SSADM*', Pitman