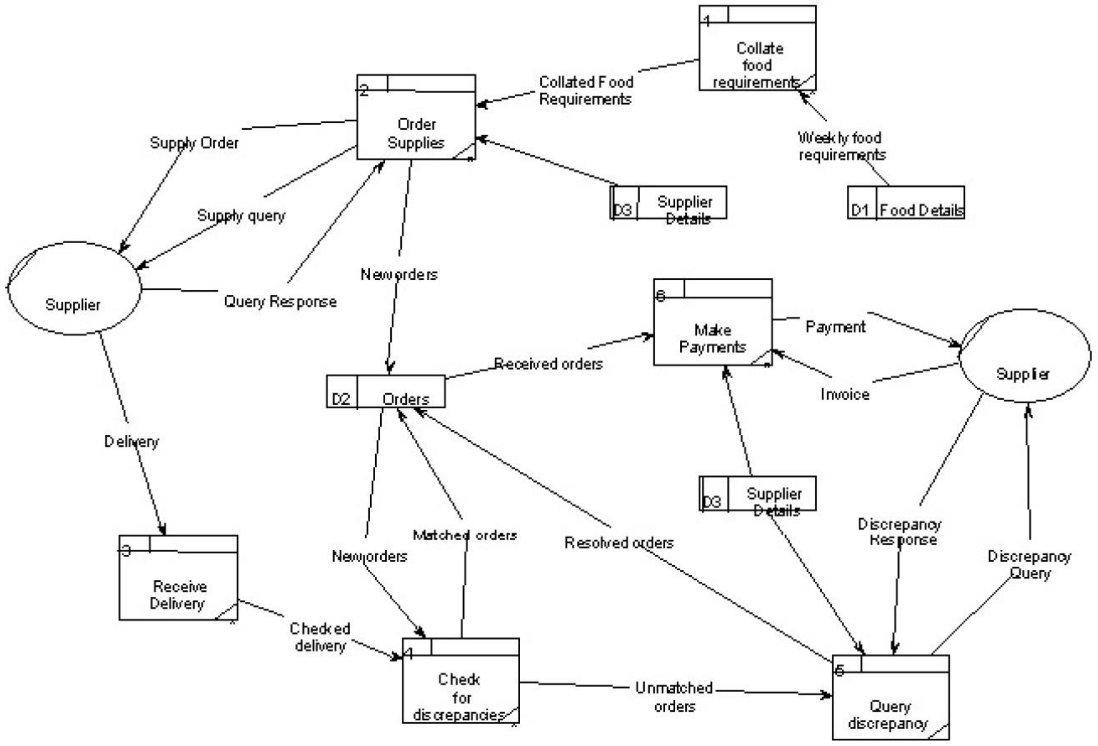
5 DATA-FLOW DIAGRAMS

<u>What are data-flow diagrams?</u>

Data-flow diagrams (DFDs) model a perspective of the system that is most readily understood by users – the flow of information through the system and the activities that process this information.
Data-flow diagrams provide a graphical representation of the system that aims to be accessible to computer specialists and non-specialist users alike. The models enable software engineers, customers, and users to work together effectively during the analysis and specification of requirements. Although this means that our customers are required to understand the modeling techniques and constructs, in data-flow modeling only a limited set of constructs are used, and the rules applied are designed to be simple and easy to follow. These same rules and constructs apply to all data-flow diagrams (i.e., for each of the different software process activities in which DFDs can be used).

An example data-flow diagram

An example of part of a data-flow diagram is given below. Do not worry about which parts of what system this diagram is describing – look at the diagram to get a feel for the symbols and notation of a data-flow diagram.



As can be seen, the DFD notation consists of only four main symbols:

1. Processes — the activities carried out by the system which uses and transform information. Processes are notated as rectangles with three parts, such as “Order Supplies” and “Make Payments” in the above example.

2. Data-flows — the data inputs to and outputs from these activities. Data-flows are notated as named arrows, such as "Delivery" and "Supply Order" in the example above.

3. External entities — the sources from which information flows into the system and the recipients of information leaving the system. External entities are notated as ovals, such as "Supplier" in the example above.

4. Data stores — where information is stored within the system. Datastores are notated as rectangles with two parts, such as "Supplier Details" and "Orders" in the example above. The diagrams are supplemented by supporting documentation including a data dictionary, describing the contents of data-flows and data stores; and process definitions, which provide detailed descriptions of the processes identified in the data-flow diagram.

<u>The benefits of the data-flow diagram</u>

Data-flow diagrams provide a very important tool for software engineering, for several reasons:

- The system scope and boundaries are indicated on the diagrams (more will be described the boundaries of systems and each DFD later in this chapter).
- The technique of decomposition of high-level data-flow diagrams to a set of more detailed diagrams provides an overall view of the complete system, as well as a more detailed breakdown and description of individual activities, where which is appropriate, for clarification and understanding.

Note
Use-case diagrams also provide a partition of a software system into those things which are inside the system and those things which are outside of the system.

Case study
We shall be using the following case study to explore different aspects of data-flow modeling and diagrams.

Video-Rental LTD case study

Video-Rental LTD is a small video rental store. The store lends videos to customers for a fee and purchases its videos from a local supplier.

A customer wishing to borrow a video provides the empty box of the video they desire, their membership card, and payment – payment is always with the credit card used to open the customer account. The customer then returns the video to the store after watching it.

If a loaned video is overdue by a day the customer's credit card is charged, and a reminder letter is sent to them. Each day after that a further card is made, and each week a reminder letter is sent. This continues until either the customer returns the video, or the charges are equal to the cost of replacing the video.

New customers fill out a form with their details and credit card details, and the counter staff give the new customer a membership card. Each new customer's form is added to the customer file.

The local video supplier sends a list of available titles to Video-Rental LTD, who decides whether to send them an order and payment. If an order is sent then the supplier sends the requested videos to the store. For each new video a new stock form is completed and placed in the stock file.

The different kinds (and levels) of data-flow diagrams

Although all data-flow diagrams are composed of the same types of symbols, and the validation rules are the same for all DFDs, there are three main types of data-flow diagrams:

- Context diagrams — context diagrams DFDs are diagrams that present an overview of the system and its interaction with the rest of the "world".

- Level 1 data-flow diagrams — Level 1 DFDs present a more detailed view of the system than context diagrams, by showing the main sub-processes and stores of data that make up the system as a whole.

- Level 2 (and lower) data-flow diagrams — a major advantage of the data-flow modeling technique is that through a technique called "leveling", the detailed complexity of real-world systems can be managed and modeled in a hierarchy of abstractions. Certain elements of any dataflow diagram can be decomposed ("exploded") into a more detailed model a level lower in the hierarchy.

During this unit, we shall investigate each of the three types of diagram in the sequence they are described above. This is both a sequence of increasing complexity and sophistication and also the sequence of DFDs that is usually followed when modeling systems. For each type of diagram we shall first investigate what the features of the diagram are, then we shall investigate how to create that type of diagram. However, before looking at particular kinds of dataflow diagrams, we shall briefly examine each of the symbols from which DFDs are composed.

Elements of data-flow diagrams

Four basic elements are used to construct data-flow diagrams:
- processes
- data-flows
- data stores
- external entities

The rest of this section describes each of the four elements of DFDs, in terms of their purpose, how the element is notated, and the rules associated with how the element relates to others in a diagram.

Notation and software

A number of different notations exist for depicting these elements, although it is only the shape of the symbols which vary in each case, not the underlying logic. This unit uses the Select SSADM notation in the description and construction of data-flow diagrams.

As data-flow diagrams are not a part of the UML specification, ArgoUML and Umbrello do not support their creation. However, Dia is free software available for both Windows and Ubuntu which does support data-flow diagrams.

Processes

Purpose

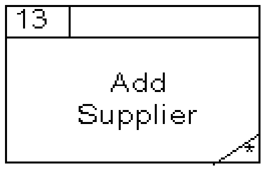
Processes are the essential activities, carried out within the system boundary, that use information. A process is represented in the model only where the information which provides the input into the activity is manipulated or transformed in some way so that the data flowing out of the process is changed compared to that which flowed in.

The activity may involve capturing information about something that the organisation is interested in, such as a customer or a customer's maintenance call. It may be concerned with recording changes to this information, a change in a customer's address for example. It may require calculations to be carried out, such as the quantity left in stock following the allocation of stock items to a customer's job; or it may involve validating information, such as checking that faulty equipment is covered by a maintenance contract.

Notation

Processes are depicted with a box, divided into three parts.

The notation for a process



The top left-hand box contains the process number. This is simply for identification and reference purposes and does not in any way imply priority and sequence.

The main part of the box is used to describe the process itself, giving the processing performed on the data it receives.

The smaller rectangular box at the bottom of the process is used in the Current Physical Data-Flow Diagram to indicate the location where the processing takes place. This may be the physical location of the Customer Services Department or the Stock Room, for example. However, it is more often used to denote the staff role responsible for performing the process. For example, Customer Services, Purchasing, Sales Support, and so on.

Rules

The rules for processes are:

- Process names should be an imperative verb specific to the activity in question, followed by a pithy and meaningful description of the object of the activity. Create Contract, or Schedule Jobs, as opposed to using very general or non-specific verbs, such as Update Customer Details or Process Customer Call.
- Processes may not act as data sources or sinks. Data flowing into a process must have some corresponding output, which is directly related to it. Similarly, data flowing out of a process must have some corresponding input to which it is directly related.
- Normally only processes that transform system data are shown on data-flow diagrams. Only where an enquiry is central to the system is it included.
- Where a process is changing data from a data store, only the changed information flow to the data store (and not the initial retrieval from the data store) is shown on the diagram.
- Where a process is passing information from a data store to an external entity or another process, only the flow from the data store to the process is shown on the diagram.

Data-flows

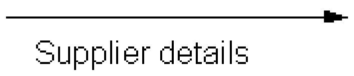
Purpose

A data flow represents a package of information flowing between two objects in the data-flow diagram. Data-flows are used to model the flow of information into the system, out of the system, and between elements within the system.

Occasionally, a data flow is used to illustrate information flows between two external entities, which is, strictly speaking, outside of the system boundaries. However, knowledge of the transfer of information between external entities can sometimes aid understanding of the system under investigation, in which case it should be depicted on the diagram.

Notation

A data flow is depicted on the diagram as a directed line drawn between the source and recipient of the data flow, with the arrow depicting the direction of flow.



The directed line is labeled with the data-flow name, which briefly describes the information contained in the flow. This could be a Maintenance Contract, Service Call Details, Purchase Order, and so on.
Data-flows between external entities are depicted by dashed, rather than unbroken, lines.

Rules

The rules for drawing data flows are:

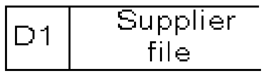
- Information always flows to or from a process; the other end of the flow may be an external entity, a data store or another process. An occasional exception to this rule is a data-flow between two external entities.
- Datastores may not be directly linked by data flows; information is transformed from one stored state to another via a process.
- Information may not flow directly from a data store to an external entity, nor may it flow from an external entity directly to a data store. This communication and receipt of information stored in the system always take place via a process.
- The sources (where data of interest to the system is generated without any corresponding input) and sinks (where data is swallowed up without any corresponding output) of data-flows are always represented by external entities.
- When something significant happens to a data flow, as a result of a process acting on it, the label of the resulting data flow should reflect its transformed status. For example, “Telephoned Service Call” becomes “Service Call Form” once it has been logged.

Data stores

Purpose

A datastore is a place where data is stored and retrieved within the system. This may be a file, Customer Contracts file for example, a catalog or reference list, Options Lists for example, a log book such as the Job Book, and so on.

Notation
A datastore is represented in the data-flow diagram by a long rectangle, containing two locations.



The small left-hand box is used for the identifier, which comprises a numerical reference prefixed by a letter.

The main area of the rectangle is labeled with the name of the data store. Brief names are chosen to reflect the content of the data store.

Rules

The rules for representing data stores are:

- One convention that could be used is to determine the letter identifying a data store by the store's nature.
- “M” is used where a manual data store is being depicted.
- “D” is used where it is a computer-based data store.
- “T” is used where a temporary datastore is being represented.
- Datastores may not act as data sources or sinks. Data flowing into a data store must have some corresponding output and vice versa.
- Because of their function in the storage and retrieval of data, data stores often provide input dataflows to receive output flows from several processes. For the sake of clarity and to avoid crisscrossing of data flows in the data-flow diagram, a single data store may be included in the diagram at more than one point. Where the depiction of a data store is repeated in this way, this is signified by drawing a second vertical line along the left-hand edge of the rectangle for each occurrence of the data store.

External entities

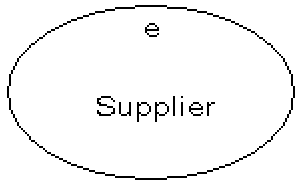
Purpose

External entities are entities outside of the system boundary which interact with the system, in that they send information into the system or receive information from it. External entities may be external to the whole organization — as in Customer and Supplier in our running example; or just external to the application area where users' activities are not directly supported by the system under investigation. Accounts and Engineering are shown as external entities as they are recipients of information from the system. Sales also provide input to the system.

External entities are often referred to as sources and sinks. All information represented within the system is sourced initially from an external entity. Data can leave the system only via an external entity

Notation

External entities are represented on the diagram as ovals drawn outside of the system boundary, containing the entity name and an identifier.



Names consist of a singular noun describing the role of the entity. Above the label, a lower case letter is used as the identifier for reference purposes.

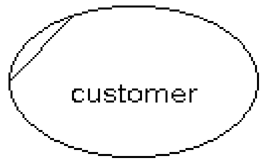
Rules

The rules associated with external entities are:

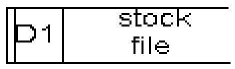
- Each external entity must communicate with the system in some way, thus there is always a data flow between an external entity and a process within the system.
- External entities may provide and receive data from several processes. It may be appropriate, for the sake of clarity and to avoid crisscrossing data flows, to depict the same external entity at a number of points on the diagram. Where this is the case, a line is drawn across the left corner of the ellipse, for each occurrence of the external entity on the diagram. Customer is duplicated in this way in our example.

Multiple copies of entities and data stores on the same diagram

At times a diagram can be made much clearer by placing more than one copy of an external entity or datastore in different places — this can avoid a tangle of crossing data-flows.
Where more than one copy of an external entity appears on a diagram it has a cut off corner in the top left, such as below:



When more than one copy of a data store appears on a diagram it has a cut-off left side, such as below:

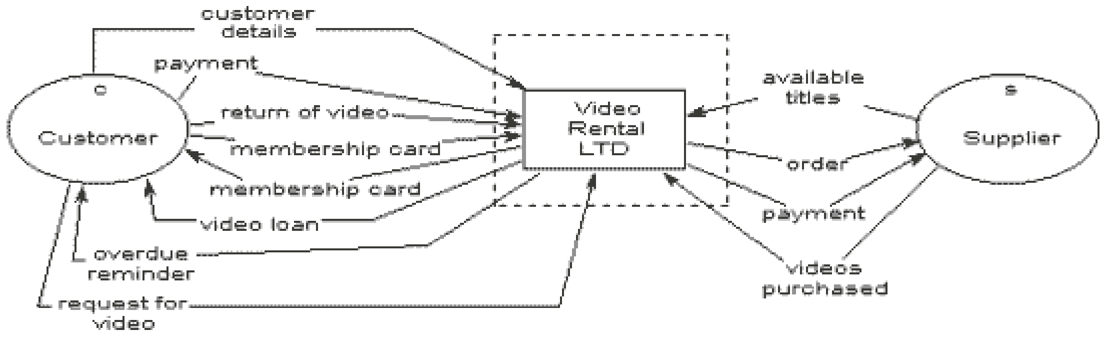


Context diagrams

What is a context diagram?
The context diagram is used to establish the context and boundaries of the system to be modelled: which things are inside and outside of the system being modeled, and what is the relationship of the system with these external entities.

A context diagram sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.

A possible context diagram for the Video-Rental LTD case study is shown below.



The process of establishing the analysis framework by drawing and reviewing the context diagram inevitably involves some initial discussions with users regarding problems with the existing system and the specific requirements for the new system. These are formally documented along with any specific system requirements identified in previous studies.

Having agreed on the framework, a detailed investigation of the current system must be planned. This involves identifying how each of the areas included within the scope will be investigated. This could be by interviewing users, providing questionnaires to users or clients, studying existing system documentation and procedures, observation and so on. Key users are identified and their specific roles in the investigation are agreed upon.

Constructing a context diagram

To produce the context diagram and agree on the system scope, the following must be identified:

- external entities
- data-flows

You may find the following steps useful:

1. Identify data flows by listing the major documents and information flows associated with the system, including forms, documents, reference material, and other structured and unstructured information (emails, telephone conversations, information from external systems, etc.).
2. Identify external entities by identifying sources and recipients of the data-flows, which lie outside of the system under investigation. The actors and any use case models you have created may often be external entities.
3. Draw and label a process box representing the entire system.
4. Draw and label the external entities around the outside of the process box.
5. Add the data flow between the external entities and the system box. Where documents and other packets of information flow entirely within the system, these should be ignored from the point of view of the context diagram – at this stage, they are hidden within the process box.

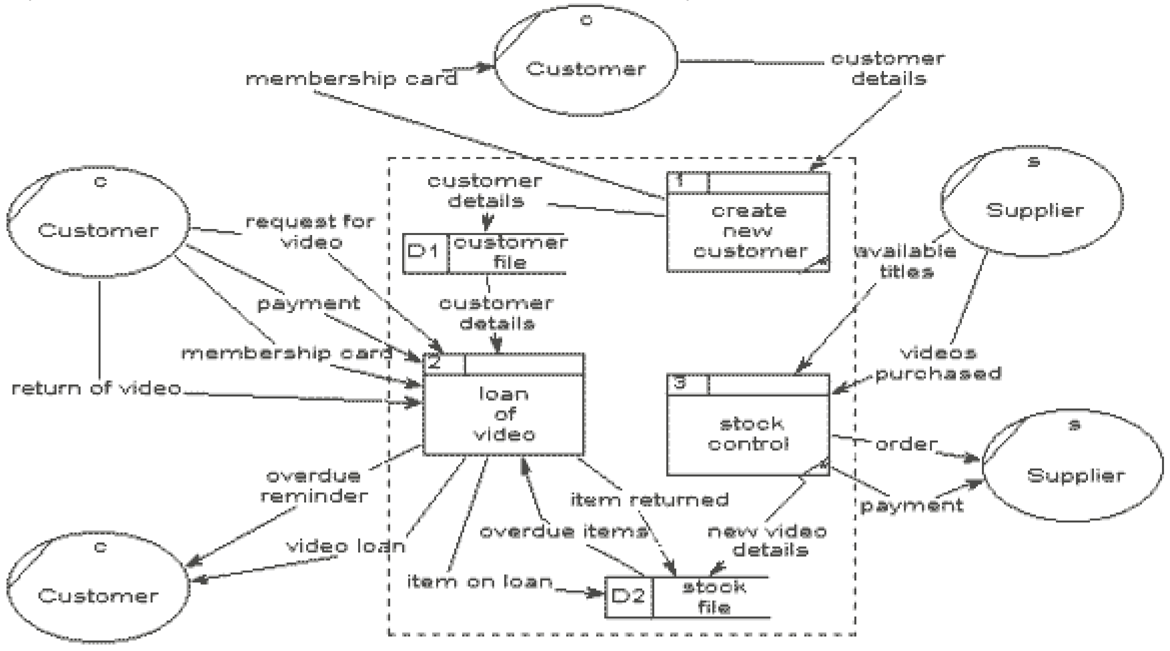
This system boundary and details depicted in the context diagram should then be discussed (and updated if necessary) in consultation with your customers until an agreement is reached.

Having defined the system boundary and scope, the areas for investigation will be determined, and appropriate techniques for investigating each area will need to be decided.

Level 1 data-flow diagrams

What is a level 1 DFD?
As described previously, context diagrams (level 0 DFDs) are diagrams where the whole system is represented as a single process. A level 1 DFD notates each of the main sub-processes that together form the complete system. We can think of a level 1 DFD as an "exploded view" of the context diagram.

A possible level 1 DFD for the Video-Rental LTD case study is as follows:



Notice that the external entities have been included in this diagram, but outside of the rectangle represents the boundary of this diagram (i.e., the system boundary). It is not necessary to always show the external entities on level 1 (or lower) DFDs, however, you may wish to do so to aid clarity and understanding.

We can see that on this level 1 DFD there are several data stores and data-flows between processes and the data stores.

It is important to notice that the same data flows to and from the external entities appear on this level 1 diagram and the level 0 context diagram. Each time a process is expanded to a lower level, the lower level diagram must show all the same data flows into, and out of the higher level process it expands.

Constructing level 1 DFDs

If no context diagram exists, first create one before attempting to construct the level 1 DFD (or construct the context diagram and level 1 DFD simultaneously). The following steps are suggested to aid the construction of Level 1 DFD:

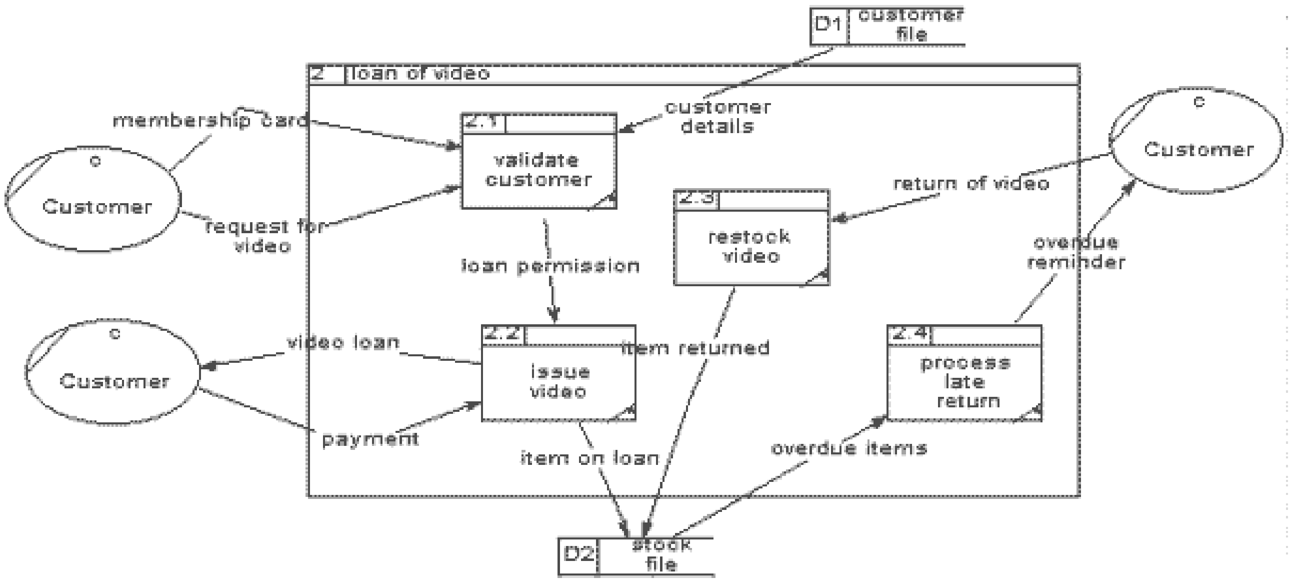
1. Identify processes. Each data flow into the system must be received by a process. For each data flow into the system examine the documentation about the system and talk to the users to establish a plausible process of the system that receives the data flow. Each process must have at least one output data flow. Each output data-flow of the system must have been sent by a process; identify the processes that send each system output.
2. Draw the data flows between the external entities and processes.
3. Identify data stores by establishing where documents/data need to be held within the system. Add the data stores to the diagram, labeling them with their local name or description.
4. Add data-flows flowing between processes and data stores within the system. Each data store must have at least one input data flow and one output data flow (otherwise data may be stored, and never used, or a store of data must have come from nowhere). Ensure every datastore has input and output data-flows to system processes. Most processes are normally associated with at least one data store.
5. Check diagram. Each process should have an input and an output. Each data store should have an input and an output. Check the system details to see if any process appears to be happening for no reason (i.e., some "trigger" data-flow is missing, which would make the process happen).

<u>Decomposing diagrams into level 2 and lower hierarchical levels</u>

What is a level 2 (or lower) DFD?
We have already seen how a level 0 context diagram can be decomposed (exploded) into a level 1 DFD. In DFD modeling terms we talk of the context diagram as the "parent" and the level 1 diagram as the "child".

This same process can be applied to each process appearing within a level 1 DFD. A DFD that represents a decomposed level 1 DFD process is called a level 2 DFD. There can be a level 2 DFD for each process that appears in the level 1 DFD.

A possible level 2 DFD for process “2: Loan of video” of the level 1 DFD is as follows:



Note, that every data flow into and out of the parent process must appear as part of the child DFD. The numbering of processes in the child DFD is derived from the number of the parent process – so all processes in the child DFD of process 2, will be called 2.X (where X is the arbitrary number of the process on the level 2 DFD). Also there are no new data-flows into or out of this diagram – this kind of data-flow validation is called balancing.

Look at the rectangular boundary for this level 2 DFD. Outside the boundary is the external entity “Customer”. Also outside the boundary are the two data stores – although these data stores are inside the system (see the level 1 DFD), they are outside the scope of this level 2 DFD.

Constructing level 2 (and lower) DFDs — functional decomposition

The level 1 data-flow diagram provides an overview of the system. As the software engineers' understanding of the system increases it becomes necessary to expand most of the level 1 processes to a second or even third level in order to depict the detail within it. Decomposition is applied to each process on the level 1 diagram for which there is enough detail hidden within the process. Each process on the level 2 diagrams also needs to be checked for possible decomposition, and so on.

A process box that cannot be decomposed further is marked with an asterisk in the bottom right-hand corner. A brief narrative description of each bottom-level process should be provided with the data flow diagrams to complete the documentation of the data-flow model. These make up part of the process definitions which should be supplied with the DFD.

Each process on the level 1 diagram is investigated in more detail, to give a greater understanding of the activities and data flows. Normally processes are decomposed where:

- There are more than six data flows around the process
- The process name is complex or very general which indicates that it incorporates a number of activities.

The following steps are suggested to aid the decomposition of a process from one DFD to a lower level DFD. As you can see they are very similar to the steps for creating a level 1 DFD from a context diagram:

1. Make the process box on the level 1 diagram the system boundary on the level 2 diagram that decomposes it. This level 2 diagram must balance with its “parent” process box — the dataflows to and from the process on the level 1 diagram will all become data flows across the system boundary on the level 2 diagram. The sources and

recipients of data-flows across the level 2 system boundary are drawn outside the boundary and labeled exactly as they are on the level 1 diagram. Note that these sources and recipients may be data stores as well external entities or other processes.This is because a data store in a level 1 diagram will be outside the boundary of a level 2 process that sends or receives data flows to/from the data store.

2. Identify the processes inside the level 2 system boundary and draw these processes and their data-flows. Remember, each data flow into and out of the level 2 system boundary should be to/ from a process. Using the results of the more detailed investigation, filter out and draw the processes at the lower level that send and receive information both across and within the level 2 system boundary. Use the level numbering system to number sub-processes so that, for example, process 4 on the level 1 diagram is decomposed to sub-processes 4.1, 4.2, 4.3 … on the level 2 diagram.
3. Identify any data stores that exist entirely within the level 2 boundary, and draw these data stores.
4. Identify data flows between the processes and data stores that are entirely within the level 2 system boundary. Remember, every data store inside this boundary should have at least one input and one output data flow.
5. Check the diagram. Ensure that the level 2 data-flow diagram does not violate the rules for dataflow diagram constructs.