Guidelines for drawing a DFD

1. Make a list of business activities: to start a DFD, collapse the organization's system narration into a list using the four basic symbols: external entity, data flow, process, data store.
2. Identify external entities by scanning the narrative.
3. Identify the inputs and outputs that are expected in the normal course of business.
4. Identify the necessary information needed in the data flow.
5. Create the context diagram:
   (a) The context diagram should be an overview including the basic inputs, the general system and outputs.
   (b) The context diagram is the most general diagram (highest level in DFD) and it includes only one process that represents the entire system and is given the number zero.
   (c) All external entities are shown on the context diagram as well as major data flows to and from them.
   (d) It does not contain data stores.
6. Create diagram 0;
   (a) More detailed diagram, achieved by exploding the context diagram.
   (b) Inputs and outputs specified in the context diagram remain constant in all consequent diagrams; however, they show data stores and new lower level data flows.
   (c) Processes are numbered with integers, generally starting from the upper left-hand corner working towards the lower right-hand corner.
   (d) Major data stores and all external entities are shown in diagram 0.
7. Draw the first draft free-hand and concentate on getting everything down except for errors, exceptions and decisions.
8. Accept that you will need at least 3 drafts for the high-level data flow.
9. Once you have the first draft, check back with your list of inputs and outputs to ensure you have everything down except those that deal with errors and exceptions.
10. Now, work on a clearer draft with a template for symbols. You are aiming at a diagram with the minimum number of crossing data flows. To reduce crossing data flows:
    (a) First, repeat external entities if necessary,

(b) then repeat data stores if necessary.

(c) finally, allows for crossing if you can't imagine a layout that reduces data flow crossing.

11. With the user or someone who's familiar with the application, conduct a walk through of the draft noting that it's just a draft and noting changes that may result from the walk-through.

12. Create child diagrams:

    (a) Processes on diagram 0 may in turn be exploded into more detailed child diagrams that deal with errors, exceptions and incorporate changes in higher-level diagrams (diagram 0) if necessary.

    (b) The process that is exploded into a more detailed diagram is called a parent process and the resulting diagram is called a child diagram.

    (c) The primary rule for creating child diagrams is that child diagrams can not produce outputs or receive inputs that the parent process does not also produce or receive. All data flows entering and leaving the parent process must be shown in the child diagram.

    (d) Child diagrams have the same numbers as their parent processes, and decimal point numbers are given to each child process.

    (e) Child diagrams can have data stores not shown on the parent process.

    (f) Entities are usually not shown on context diagrams.

    (g) Sometimes it is confusing to know how many processes to place into one diagram and when to create child diagrams. A good suggestion is to examine each process and count data flows entering and leaving a process, if the count is greater than 4 then the process is good candidate for a child diagram.

13. Check the diagrams for errors in drawing DFD:

    (a) Forgetting to include a data flow or pointing an arrow in the wrong direction.

    (b) Connecting a data store or external entity directly to each other. They must be connected only with a process.

    (c) Incorrectly labelling a processing and/or a data flow.

    (d) Having more than 9 processes in the diagram can create a confusing data flow diagram. If more than 9 processes are involved in the system, group some of the processes that work together into a subsystem and place them into a child diagram.

    (e) Omitting data flow. Examine your diagram for linear flow (each process has one input and one output), linear flow is rare and its presence indicates your diagram has missing data flows.

(f) Creating an unbalanced decomposition into child diagrams. Child diagrams must have the same inputs and outputs as their parent processes.

14. Logical and physical DFDs:

    (a) Logical DFDs focus on the business, how the business operates and the business activities that take place and the data required and produced by each event. They are not concerned with how the system is constructed.

    (b) Physical DFDs show how the business is implemented including hardware, software, files and people involved in the system.

15. **Partitioning:**

    (a) Partitioning is the process of examining a DFD and determining how it should be divided into a collection of manual procedures and computer programs.

    (b) Automated procedures should be grouped into a series of computer programs.

    (c) Dashed lines around a process or group of processes means that they should be grouped into a single computer program.

    (d) Automated processes become either batch or on-line.

    (e) If data flows entering and leaving a process consist entirely of stored information generated and accessed by computers, thus requiring no human interaction then the process is batch.

    (f) If data flows connect a manual process or external entity to an automated process, it represents a person-computer interaction requiring a user interface.

    (g) Batch processes are usually used when the programs process a high volume of data.

    (h) On-line processes are used when programs process a low volume of transactions and inquiries.

16. Why partitioning:

    (a) **Different user groups:** are processes performed by different user groups, often at different physical locations at the same company, then they should be partitioned into different program.

    (b) **Timing:** Examine the timing of processes. If processes execute at different times, then they should not be grouped into a single program.

    (c) **Similar tasks:** if two processes perform similar tasks and are both batch..

    (d) **Efficiency**

(e) **Consistency of data**
(f) **Security.**